Kauno technologijos universitetas

Informatikos fakultetas

UML panaudojimo atvejų diagramų eskizų skaitmeninimas

Baigiamasis magistro projektas

Benas Miliūnas

Projekto autorius

doc. Mantas Jurgelaitis

Vadovas

Kaunas, 2024

Kauno technologijos universitetas

Informatikos fakultetas

UML panaudojimo atvejų diagramų eskizų skaitmeninimas

Baigiamasis magistro projektas

Veiklos skaitmeninimas ir sistemų architektūros (6211BX009)

Benas Miliūnas

Projekto autorius

doc. Mantas Jurgelaitis

Vadovas

prof. Evaldas Vaičiukynas

Recenzentas

Kaunas, 2024

Kauno technologijos universitetas

Informatikos fakultetas

Benas Miliūnas

UML panaudojimo atvejų diagramų eskizų skaitmeninimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Benas Miliūnas

Patvirtinta elektroniniu būdu

Miliūnas, Benas. UML panaudojimo atvejų diagramų eskizų skaitmeninimas. Magistro projektas / vadovas. doc. Mantas Jurgelaitis; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informacijos sistemos, Informatikos mokslai.

Reikšminiai žodžiai: skaitmenizavimas; automatizavimas; PA diagrama, UML.

Kaunas, 2024. 91 p.

Santrauka

Šiame darbe yra pristatomas UML panaudojimo atvejų diagramų skaitmenizavimo įrankis iš eskizo. Atlikta UML kalbos ir jos taikymo sistemų projektavime analizė parodė, kad pirmieji diagramų eskizai neretai nubraižomi ranka be CASE įrankio. Atlikus sprendimų analizę buvo suprasta, kad nėra visapusiško įrankio, kuris padėtų skaitmenizuoti PA diagramos eskizą ir suteiktų galimybę pradinio eskizo modeliavimą pratęsti CASE įrankyje. Įrankiui įgyvendinti buvo sudarytas naujas PA diagramų rinkinys iš 346 suanotuotų eskizų bei išanalizuotos mašininio mokymosi galimybės vaizdų ir teksto atpažinimui. Diagramos simbolių atpažinimo modeliui pasirinkta naudoti *Faster R-CNN ResNet152 V1 800x1333* neuroninio tinklo architektūrą *TensorFlow* 2.5 ekosistemoje, o teksto atpažinimui *Google Cloud Vision API* išorinę paslaugą. Suprojektuotas įrankis realizuotas ir sudarytas iš 5 pagrindinių komponentų su *Python* programavimo kalba: diagramos simbolinių atpažinimo komponento, raktinių taškų nustatymo komponento, OCR komponento, simbolių susiejimo ir diagramos konvertavimo į XMI failą komponentų. Galiausiai buvo atliktas diagramos simbolių atpažinimo ir diagramos skaitmenizavimo eksperimentai su 32 naujų PA diagramų. Diagramos simbolių atpažinimo eksperimentas parodė, kad elementų atkūrimas (angl. *recall*) yra 0.96, preciziškumas 0.91 (angl. *precision*), o F1 statistikos įvertis – 0.94. Diagramos skaitmenizavimo eksperimentas parodė, kad diagramos elementai skaitmenizuojami 0.93 atkūrimu, 0.97 preciziškumu ir F1 statistikos įverčiu 0.95, o sujungiami 0.84 atkūrimu su 0.95 preciziškumu ir 0.89 F1 statistikos įverčiu.

Miliunas, Benas. Digitalization of UML Use Case Diagram Sketches. Master's Final Degree Project / supervisor assoc. prof. Mantas Jurgelaitis; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Information Systems, Computing.

Keywords: digitisation; automation; use case diagram, UML.

Kaunas, 2024. 91 p.

Summary

This paper presents a tool for digitising UML use case diagrams from a sketch. An analysis of the UML language and its application in systems design has shown that first sketches of diagrams are often drawn by hand without the CASE tool. The solution analysis showed that there is no comprehensive tool to digitise use case diagram sketch and to allow the modelling of the initial sketch to be extended in the CASE tool. To implement the tool, a new set of use case diagrams was created from 346 annotated sketches and machine learning capabilities for image and text recognition were analysed. For the diagram elements recognition model, the *Faster R-CNN ResNet152 V1 800x1333* neural network architecture in the TensorFlow 2.5 ecosystem was chosen, and for the text recognition, an external service of the *Google Cloud Vision API* was used. The designed tool is implemented and composed of 5 main components with Python programming language: a diagram symbol recognition component, a keypoint detection component, an OCR component, a symbol linking component and a diagram to XMI file conversion component. Finally, an experiment on diagram symbol recognition and diagram digitisation was carried out with 32 new diagrams. Diagram symbol recognition experiment showed a recall of 0.98, precision of 0.92 and F1-score of 0.94. Diagram digitisation experiment showed that diagram elements are digitised with a recall of 0.96, precision of 0.99 and F1 score of 0.95. Symbols connection was achieved with a recall of 0.89, a precision of 0.99 and F1-score of 0.89.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

UML – Vieninga modeliavimo kalba (angl. *Unified Modelling Language*).

PA – Panaudojimo atvejų diagrama (angl. *Use Case*).

SVM – Atraminių vektorių mašina (angl. *Support Vector Machine*).

CNN – Konvoliuciniai neuroniniai tinklai (angl. *Convolutional Neural Networks*).

DNN – Gilieji neuroniniai tinklai (angl. *Deep Neural Networks*).

RNN – Rekurentiniai neuroniniai tinklai (angl. *Recurrent Neural Networks*).

XMI – (angl. *XML Metadata Interchange*) XML meta kalba pagrįstas metamodelių apsikeitimo formatas.

XML – (angl. *Extensible Markup Language*) bendros paskirties duomenų struktūrų aprašomoji kalba.

Įvadas

Vieningos modeliavimo kalbos UML (angl. *Unified Modeling Language)* diagramos yra priemonė naudojama apibrėžti esamų arba planuojamų kurti sistemų struktūrą ir elgseną [1]. Tokiomis diagramomis galima pateikti sistemos abstrakciją, nesigilinant į realizaciją. Kadangi UML notacija yra standartizuota, tai įgalina komunikaciją tarp skirtingų sričių specialistų ankstyvuose sistemos kūrimo etapuose, neapsiribojant konkrečiais sprendimo būdais, įrankiais ir technologijomis. Nors šiuo metu rinkoje egzistuoja daug įrankių, leidžiančių braižyti skaitmenizuotas UML diagramas, praktikoje dažnai pirmieji diagramų eskizai yra nubraižomi ranka ant popieriaus arba išmaniosios lentos (angl. *white board*), diskusijų metu tarp sistemos kūrėjų ir užsakovo [2]. Norint efektyviai pasidalinti eskizu su kitais interesuotais asmenimis bei toliau plėtoti pradinį diagramos eskizą, pastarąjį reikia rankiniu būdu perkelti į UML standartą palaikantį įrankį. Verta paminėti, kad UML modeliai gali būti saugomi XMI formatu, kurį gali apdoroti skirtingi įrankiai. Rankinis UML diagramų kūrimas gali būti vykdomas ir akademinėje srityje, nes taip lengviau ir paprasčiau studentams nubraižyti nurodytas diagramas atsiskaitymų metu. Dabar atsiskaitymų egzemplioriai yra neskaitmenizuojami, nes nėra patogaus būdo kaip tai padaryti. Taigi, rankinis eskizų skaitmenizavimas yra neefektyvus ir užtrunkantis daug laiko. Šį procesą galima patobulinti automatiškai sugeneruojant skaitmenizuotą UML diagramą iš diagramos eskizo, panaudojus atitinkamus mašininio mokymosi ir optimizavimo metodus.

Ranka braižytų diagramų atpažinimo literatūroje dažniausiais skirstomas į dvi pagrindines kategorijas: realaus laiko braižomų diagramų (angl. *online-targeted*) ir eskizų (angl. *offline-targeted)* atpažinimą ir skaitmenizavimą [3]. Pirmojoje kategorijoje atpažįstamos ir skaitmenizuojamos UML diagramos, kurios yra braižomos naudojant išmaniąsias lentas, kai turimi realaus laiko istoriniai duomenys apie naudotojų atliktus braižymo veiksmus. Šioje kategorijoje sprendžiami optimizavimo uždaviniai, kuriais siekiama sugrupuoti erdvėje ir laike esančius arčiausius UML diagramos simbolius [4]. Antrojoje kategorijoje atpažįstamos ir skaitmenizuojamos ranka braižytos diagramos iš nuotraukos. Šioje kategorijoje nesaugomi laiko ir erdvės duomenys apie nubraižytą diagramą, todėl dažniausiai taikomi mašininio mokymosi, kompiuterinės regos metodai [3] [5], kuriais atpažįstami UML notacijos simboliai, o vėliau susiejami, siekiant atkurti visą diagramos struktūrą. Be UML simbolių atpažinimo ir diagramos sudarymo, taikomi nuotraukos apdorojimo, teksto atpažinimo ir diagramų klasifikavimo algoritmai [6] [7].

Šiuo metu daugiausia tyrimų yra atlikta rankiniu būdu braižytų UML diagramų eskizų automatiniam skaitmenizavimui, kai turimi laiko ir erdvės duomenys (angl. *online-targeted*) [8] [6] [4], tačiau tyrimų susijusių su UML diagramų skaitmenizavimu iš nuotraukų (angl. *offline-targeted*) yra gerokai mažiau [3]. Literatūroje pastebima, kad geriausi rezultatai yra pasiekiami kombinuojant įvairių giliųjų neuroninių tinklų (angl. *deep neural network*) architektūras, pavyzdžiui, sujungiant rekurentinį neuroninį tinklą RNN su konvoliuciniu neuroniniu tinklu CNN [3] [9].

Tikslas – automatizuoti UML panaudojimo atvejų diagramų eskizų skaitmenizavimą.

Uždaviniai:
1. Išanalizuoti UML kalbą ir jos taikymą sistemų projektavime.
2. Išanalizuoti mašininio mokymosi galimybes vaizdų ir teksto atpažinimui.
3. Išanalizuoti egzistuojančias automatinių UML ir kitų modeliavimo kalbų diagramų eskizų skaitmenizavimo sąsajas.

4. Suprojektuoti UML panaudojimo atvejų diagramų eskizų skaitmenizavimo įrankį.
5. Realizuoti UML panaudojimo atvejų diagramų eskizų skaitmenizavimo įrankį, sukuriant reikalingą mašininio mokymosi modelį.
6. Eksperimentiškai ištirti automatizuotą UML panaudojimo atvejų diagramų eskizų skaitmenizavimo įrankį.

1. Probleminė srities analizė

1.1. Analizės tikslas

Analizės tikslas - išanalizuoti su tyrimo objektu susijusias sritis: UML kalbą ir jos taikymą sistemoms projektuoti, galimus mašininio mokymosi metodus vaizdų ir teksto atpažinimui. Taip pat, išanalizuoti esamas tyrimo problemos sprendimo sąsajas ir jas palyginti. Atlikus analizę pasiūlyti sprendimą, kuris išspręstų tyrimo problemą ir būtų pranašesnis už esamus sprendimus.

1.2. Tyrimo objektas, sritis ir problema

Tyrimo problema

Įvairiose dalykinėse srityse braižomi UML diagramų eskizai rankiniu būdu. Siekiant toliau plėtoti pradinį eskizą ir patogiai pasidalinti su kitais interesuotais asmenimis, diagramos eskizas turi būti perkeltas į UML standartą palaikantį įrankį rankiniu būdu. Toks procesas yra neefektyvus, reikalaujantis daug laiko, atsiranda galimybė padaryti klaidų.

Tyrimo objektas

UML panaudojimo atvejų diagramų eskizų skaitmenizavimas

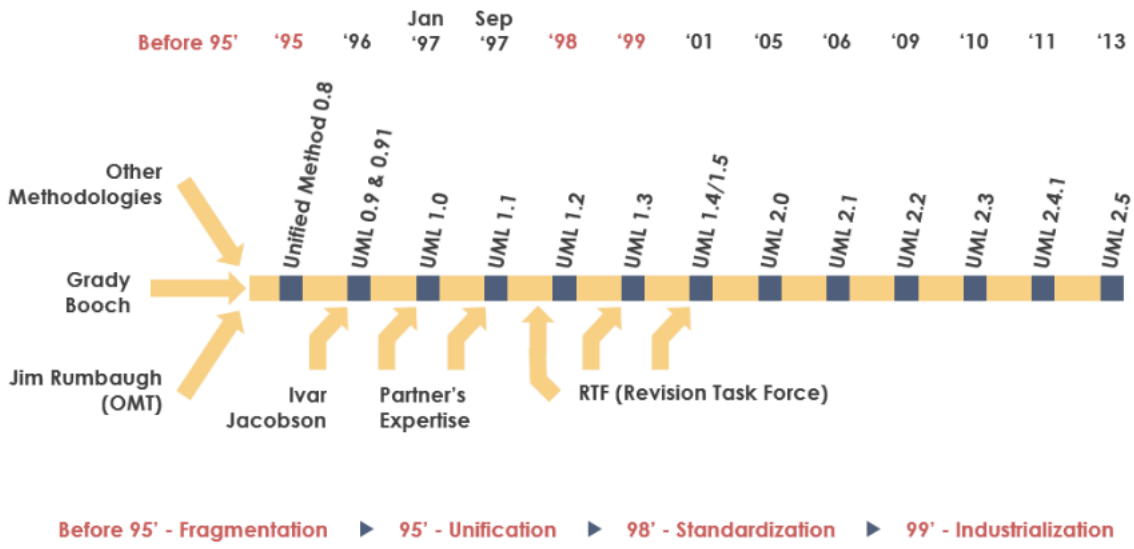
Tyrimo sritis

Mašininio mokymosi algoritmų taikymas UML panaudojimo atvejų diagramų teksto ir vaizdo atpažinimui.

1.3. Tyrimo objekto analizė

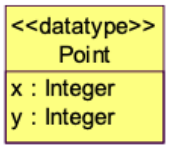
1.3.1. UML kalba ir jos taikymas sistemoms projektuoti

UML kalba yra viena iš rinkoje naudojamų modeliavimo kalbų. UML buvo sukurta ir yra palaikoma OMG (angl. *Object Management Group*) grupės, siekiant išspręsti nesuderinamumo ir standarto trūkumo problemas, kai tuo metu (1980 m. – 1990 m.) egzistavo daug skirtingų modeliavimo notacijų [1] [10] skirtingiems panaudojimo atvejams. OMT (angl. *Object Modelling Technique*) metodika buvo plačiai naudojama sistemų analizei, kurios turėjo apdoroti didelius duomenų kiekius. Booch notacija gerai tiko atspindėti sistemų architektūrą ir realizaciją, OOSE (angl. *Object-Orentied Software Engineering*) pateikė panaudojimo atvejų (angl. *Use Case*) modelį, kuris tiko pavaizduoti bendram sistemos funkcionalumui tarp skirtingų aktorių [11]. Didelis kiekis skirtingų modeliavimo galimybių ir aiškaus standarto nebuvimas kėlė sunkumų projektuotant sistemas, procesus, nes patirtis su įvairiais įrankiais, žinios negalėjo būti lengvai panaudojamos skirtinguose projektuose [10]. Taip pat, atlikta dokumentacija galėjo tapti nebenaudinga, jeigu nepopuliari modeliavimo kalba rinkoje taptų nebenaudojama [10]. Rinkoje plačiai naudojamų modelių kūrėjų susivienijimas ir bendradarbiavimas įgalino pirmojo UML standarto *Unified Method 0.8* sukūrimą 1996 m., kuris sujungė naudingiausius tuo metu rinkoje egzistuojančius modelius ir pasiūlė naujų modelio kūrimo galimybių [11].



1.1 pav. UML kalbos raida [11]

UML yra standartizuota modeliavimo kalba, turinti apibrėžtų simbolių ir diagramų rinkinius, kuriais galima projektuoti sistemų architektūrą ir elgseną bei modeliuoti įvairius verslo procesus [1]. UML turi išsamią specifikaciją, kurioje aprašyti UML modeliavimo principai, simbolių ir diagramų semantinė reikšmė bei kaip skirtingi elementai turėtų būti naudojami kartu [1]. Standartinis UML simbolių paketas gali būti pritaikytas prie konkrečių įmonių, sistemų, modeliuojamų procesų, praplečiant egzistuojančių UML elementų metamodelius [12]. UML metamodeliai gali būti konvertuojami į XMI formatą (1.2 pav., 1.3 pav.) ir naudojami bet kokiame įrankyje, kuris realizuoja pateiktą UML specifikaciją. XMI yra plačiai naudojamas apsikeitimo formatas, siekiant pasidalinti sukurtus modelius su įvairiomis šalimis ir yra realizuotas XML [13] metakalba [14]. Kaip ir UML kalba, XMI formatas turi apibrėžtą specifikaciją [14]. XMI specifikacijoje yra nurodoma, kaip įvairūs UML elementai yra atvaizduojami XML formatu. XMI formatas apibrėžia ne tik pavienių UML elementų atvaizdavimą XML formatu, bet ir mechanizmą, kaip elementai turėtų būti susiejami modelyje. Be to, XMI formatas įgalina automatinį modelių validavimą panaudojant egzistuojančias XML validavimo schemas. Tai leidžia greičiau patikrinti modelio sintaksės korektiškumą, tačiau automatinis validavimas neužtikrina modelio kokybės [14]. Verta paminėti, kad praplėtus UML elementų metamodelius, gali reikėti praplėsti ir XMI generavimo ir importavimo galimybes įrankyje, laikantis nurodytos XMI specifikacijos [1] [14].



1.2 pav. UML duomenų tipo elementas grafine notacija [1]

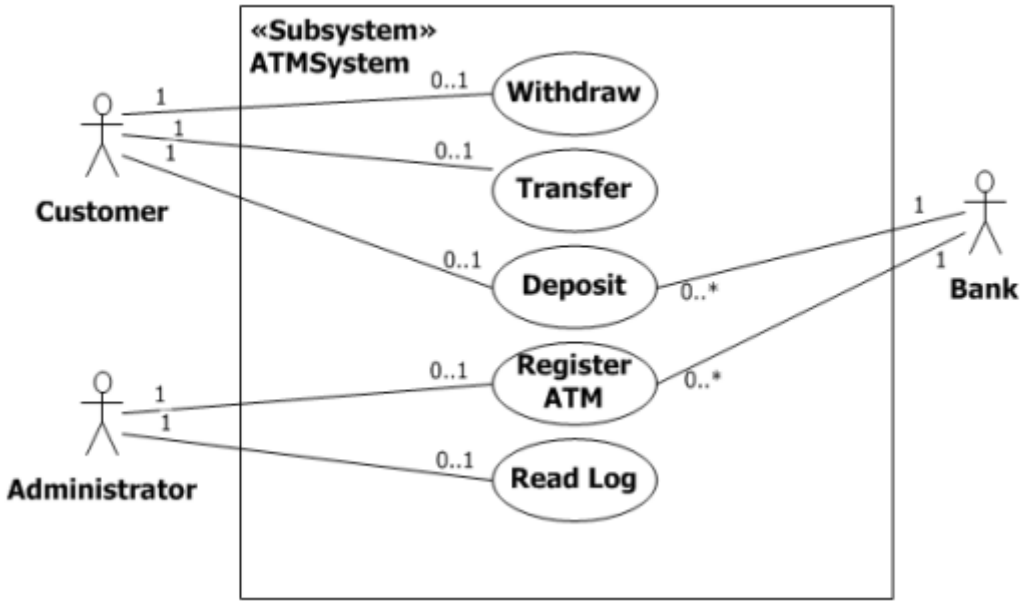
```
<g:Graph xmi:type="g:Graph">
    <points xmi:type="g:Point" x="0" y="0"/>
    <points xmi:type="g:Point" x="1" y="5"/>
</g:Graph>
```

1.3 pav. UML duomenų tipo elementas XMI formatu [1]

Kadangi UML notacija yra standartizuota, tai įgalina komunikaciją tarp skirtingų sričių specialistų ankstyvuose projektų vystymo etapuose, neapsiribojant konkrečiais sprendimo būdais ir technologijomis. Šiuo metu rinkoje egzistuoja daugiau nei 40 skirtingų įrankių palaikančių UML 2 specifikaciją ir leidžiančių atlikti UML modelio importavimą iš XMI formato ir UML modelio generavimą į XMI formatą [15]. Kurti UML modelius galima ir nenaudojant jokių įrankių. Neretai pirmieji diagramų eskizai ir yra nubraižomi ranka ant popieriaus arba išmaniosios lentos (angl. *white board*), diskusijų metu tarp sistemos kūrėjų ir užsakovo [2], o vėliau gali būti perkeliami į UML standartą palaikantį įrankį, tolimesniam plėtojimui ar patogiam pasidalinimui su kitomis interesuotomis šalimis. Rankinis UML diagramų kūrimas gali būti vykdomas ir akademinėje srityje, nes taip lengviau ir paprasčiau studentams nubraižyti nurodytas diagramas atsiskaitymų metu.

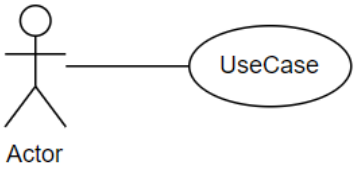
1.3.1.1. Panaudojimo atvejų diagrama

Panaudojimo atvejų (PA) diagrama (1.4 pav.) yra viena iš UML specifikacijoje aprašytų diagramų, kuri naudojama kaip sistemos funkcinių reikalavimų dokumentavimo metodas [1]. Pagrindiniai PA diagramos komponentai yra: aktoriai, panaudojimo atvejai ir ryšiai. Aktoriai gali atspindėti sistemos naudotoją arba gali būti naudojami kaip išorinės sistemos, komponentai. Panaudojimo atvejų elementai nusako sistemos elgseną, neprisirišant prie konkrečių realizacijos detalių. Ryšiai diagramoje apibrėžia sąveikas tarp diagramos elementų: aktorių ir panaudojimo atvejų. Ryšių galuose galima nurodyti daugialypiškumą (angl. *Multiplicity*). Jeigu daugialypiškumas yra didesnis nei 1 prie aktoriaus elemento galo, tai reiškia, kad panaudojimo atvejui įgyvendinti reikia daugiau negu vieno to paties tipo aktoriaus.

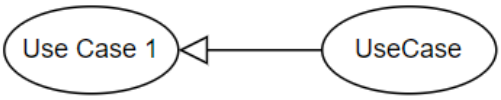


1.4 pav. Panaudojimo atvejų diagramos pavyzdys [1]

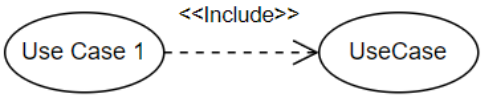
Pagrindiniai ryšio tipai naudojami PA diagramose yra: asociacija (1.5 pav.), paveldėjimo ryšys (1.6 pav.) bei priklausomybės ryšys. Pastarasis yra išskaidytas į apėmimo (1.7 pav.) ir išplėtimo (1.8 pav.) ryšius. Šie ryšiai pasižymi skirtingomis savybėmis, semantine prasme ir apribojimais.



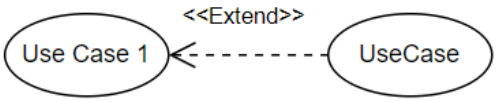
1.5 pav. Asociacijos ryšiu sujungti elementai



1.6 pav. Paveldėjimo ryšiu sujungti elementai



1.7 pav. Priklausomybės – apėmimo ryšiu sujungti elementai



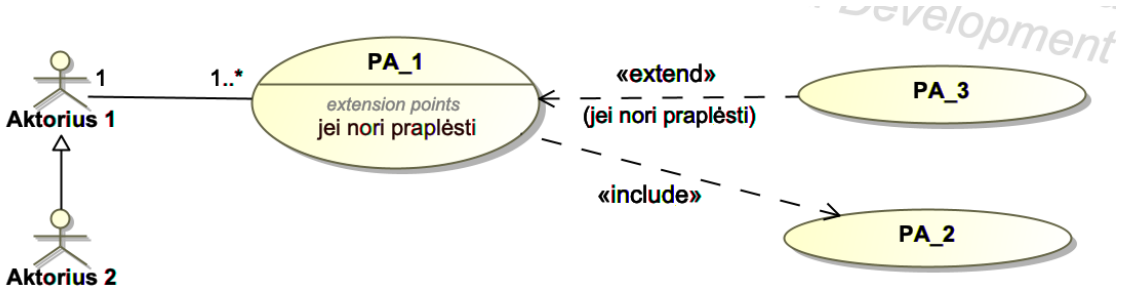
1.8 pav. Priklausomybės – išplėtimo ryšiu sujungti elementai

Asociacijos ryšiu sujungti panaudojimo atvejų ir aktoriaus elementai nusako, kad sistemos aktorius gali atlikti panaudojimo atvejo elementu nusakytą elgseną. Svarbu paminėti, kad asociacijos ryšiu negali tarpusavyje būti sujungti to paties tipo elementai, pavyzdžiui, du panaudojimo atvejai, arba du aktoriai, nes tai neturi jokios semantinės prasmės. Priešingai, paveldėjimo ryšiu gali būti sujungti tik to pačio tipo elementai, t.y. aktorius su aktoriumi, arba panaudojimo atvejis su panaudojimo atveju. Paveldėjimo ryšiu yra nusakoma, kad vienas elementas, be savo turimų savybių, paveldi kito elemento savybes. Priklausomybės ryšiais tarpusavyje gali būti siejami tik panaudojimo atvejų elementai. Išplėtimo (angl. e*xtend*) ryšys praplečia bazinę panaudojimo atvejų elgseną nurodytomis sąlygomis. Kur apibrėžti išplėtimo ryšio sąlygas diagramoje nėra specifikuota, bet vienas iš galimų variantų yra nurodyti išplėtimo sąlygą prie ryšio vidurio. Apėmimo ryšiu sujungti elementai reiškia, kad vykdant vieną panaudojimo atvejo elementą, būtinai bus įvykdytas ir kitas panaudojimo atvejo elementas, iš kurio yra rodomas apėmimo ryšys. Šis ryšys dažnai naudojamas norint iškelti bendrą funkcionalumo dalį iš kelių panaudojimo atvejų.

PA diagramoje panaudojimo atvejai dažniausiai žymimi elipse, o aktoriai „stick-man“ piktograma. Asociacijos ryšys žymimas nepertraukiama linija, paveldėjimo ryšys su tuščiavidure galvute ryšio gale. Priklausomybės ryšiai yra žymimi punktyrine linija bei turi rodyklės galvutę ryšio gale. Kadangi apėmimo ir išplėtimo ryšiai atrodo identiškai naudojamos žymos ***<<include>>*** ir ***<<exclude>>*** nusakyti priklausomybės ryšio tipą.

1.3.1.2. XMI generavimas panaudojimo atvejų diagramai

UML diagrama (1.9 pav.) gali būti paverčiama į XMI formatą (1.10 pav.) ir importuojama į bet kokį įrankį, kuris remiasi UML ir XMI specifikacijomis. XMI yra plačiai panaudojamas formatas modelių mainams su įvairiomis šalimis ir yra paremtas XML metakalba. Kaip ir UML kalba, XMI formatas yra aprašytas konkretaus standarto [14]. XMI specifikacija nurodo kaip įvairūs UML elementai turi būti sukuriami XML formatu. Be to, XMI formatas ne tik apibrėžia pavienių UML elementų sukūrimą XML formatu, bet ir nurodo mechanizmą kaip šie elementai turėtų būti susieti modelyje.



1.9 pav. Panaudojimo atvejų diagramos pavyzdys

```
<xmi:XMI xmlns:uml="http://www.omg.org/spec/UML/20110701" xmlns:xmi=" http://www.omg.org/spec/UML/20110701">
<uml:Package xmi:id="pa" xmi:label="PA" name="Generated UC Diagram">
    <packagedElement xmi:type='uml:Actor' xmi:id='actor_1' name='Aktorius 1'/>
    <packagedElement xmi:type='uml:UseCase' xmi:id='use_case_1' name='PA_1'>
        <extensionPoint xmi:type='uml:ExtensionPoint' xmi:id='extension_point_1' name='jei nori praplėsti' visibility='public'>
            <extension xmi:idref='extend_1'/>
        </extensionPoint>
        <include xmi:type='uml:Include' xmi:id='include_1' visibility='public' addition='use_case_2'/>
    </packagedElement>
    <packagedElement xmi:type='uml:Association' xmi:id='association_1'>
        <memberEnd xmi:idref='association_1_member_end_1'/>
        <memberEnd xmi:idref='association_1_member_end_2'/>
        <navigableOwnedEnd xmi:idref='association_1_member_end_1'/>
        <navigableOwnedEnd xmi:idref='association_1_member_end_2'/>
        <ownedEnd xmi:type='uml:Property' xmi:id='association_1_member_end_1' visibility='private' type='use_case_1' association='association_1'>
            <xmi:Extension extender='MagicDraw UML 2022x'>
                <modelExtension>
                    <lowerValue xmi:type='uml:LiteralInteger' xmi:id='association_1_member_end_1_c_lower' value='1'/>
                </modelExtension>
            </xmi:Extension>
            <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='association_1_member_end_1_c_upper' value='*'/>
        </ownedEnd>
        <ownedEnd xmi:type='uml:Property' xmi:id='association_1_member_end_2' visibility='private' type='actor_1' association='association_1'>
            <xmi:Extension extender='MagicDraw UML 2022x'>
                <modelExtension>
                    <lowerValue xmi:type='uml:LiteralInteger' xmi:id='association_1_member_end_2_c_lower' value='1'/>
                </modelExtension>
            </xmi:Extension>
            <xmi:Extension extender='MagicDraw UML 2022x'>
                <modelExtension>
                    <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='association_1_member_end_2_c_upper' value='1'/>
                </modelExtension>
            </xmi:Extension>
        </ownedEnd>
    </packagedElement>
    <packagedElement xmi:type='uml:Actor' xmi:id='actor_2' name='Aktorius 2'>
        <generalization xmi:type='uml:Generalization' xmi:id='generalization_1' general='actor_1'/>
    </packagedElement>
    <packagedElement xmi:type='uml:UseCase' xmi:id='use_case_2' name='PA_2'/>
    <packagedElement xmi:type='uml:UseCase' xmi:id='use_case_3' name='PA_3'>
        <extend xmi:type='uml:Extend' xmi:id='extend_1' visibility='public' extendedCase='use_case_1'>
            <extensionLocation xmi:idref='extension_point_1'/>
        </extend>
    </packagedElement>
</uml:Package>
</xmi:XMI>
```

1.10 pav. Panaudojimo atvejų diagramos pavyzdys XMI formatu

1.10 pav. yra pavaizduota UML panaudojimo atvejų diagrama konvertuota į XMI formato failą. XMI failas pradedamas kurti su žyme ***<xmi:XMI>***. Paketas yra sukuriamas su žyme ***<uml:Package>***, kurio viduje gali būti talpinami PA diagramos elementai. PA diagramos elementai yra kuriami su ***<packagedElement>*** žyme. Kiekvienas elementas turi atributus ***xmi:type***, ***name*** ir ***xmi:id***. ***Name*** atributas nusako elemento pavadinimą, kuris matomas diagramoje. Atributas ***xmi:id*** yra elemento unikalus identifikatorius, kuris naudojamas susieti elementus tarpusavyje. Identifikatorius susiejamas su kitu elementu per atributą ***xmi:idref***. ***Xmi:type*** žymos atributas yra skirtas nusakyti kokio tipo diagramos elementas yra kuriamas. Pagrindiniams PA diagramos elementams sukurti naudojamos ***xmi:type*** atributo reikšmės pateiktos 1.1 lentelėje

1.1 lentelė. *xmi:type* atributo galimos reikšmės

***xmi:type* atributo reikšmė**	**Elementas**
uml:Actor	Aktorius
uml:UseCase	Panaudojimo atvejis

uml:Generalization	Paveldėjimo ryšys
uml:Include	Apėmimo ryšys
uml:Extend	Išplėtimo ryšys
uml:Association	Asociacija
uml:ExtensionPoint	Praplėtimo taškas

Asociacijos tipo simboliai savyje turi žymes ***<memberEnd>***, ***<navigableOwnedEnd>*** ir ***<ownedEnd>***, kuriomis ryšių galams priskiriami diagramos simboliai pagal jų identifikatorius. ***<ownedEnd>*** žymos bloke galima sukurti ***<lowerValue>*** ir ***<upperValue>*** žymas, kuriomis nusakomas ryšio galo kardinalumas. Norint pažymėti paveldimumo ryšį naudojama ***<generalization>*** žyma elemento viduje, kuris paveldi norimo simbolio, pažymėto ***general*** atribute, savybes. Priklausomybės apėmimo ryšys yra sukuriamas su ***<include>*** žyma ir atributu ***addition***, kuris nusako įtraukiamo elemento identifikatorių. Priklausomybės išplėtimo ryšys yra sukuriamas su žyma ***<extend>*** ir atributu ***extendCase***, kuris nusako praplėčiantį panaudojimo atvejo identifikatorių. Praplėtimo taškai ir sąlygos yra sukuriami žymomis ***<extension>***, ***<extensionPoint>*** ir ***<extensionLocation>***. Pagrindinių panaudojimo atvejų elementų XMI fragmentai yra pateikiami 1.2 lentelėje

1.2 lentelė. PA diagramos elementai XMI formatu

XMI fragmentas	**Komentaras**	**Elementas**
<packagedElement xmi:type='uml:Actor' xmi:id='id' name='Actor'/>	-	Aktorius
<packagedElement xmi:type='uml:UseCase' xmi:id='id' name='Use case'/>	-	Panaudojimo atvejis
<include xmi:type='uml:Include' xmi:id='id' visibility='public' addition='id2'/>	Kuriamas praplečiamo panaudojimo atvejų elemento viduje.	Apėmimo ryšys
<generalization xmi:type='uml:Generalization' xmi:id='id' visibility='public' addition='id2'/>	Kuriamas aktoriaus elemento viduje, kuris paveldi kito aktoriaus funkcionalumą	Paveldėjimo ryšys
<extend xmi:type='uml:Extend' xmi:id='id' visibility='public' extendCase='id2'/>	Kuriamas panaudojimo atvejo elemento viduje, kuris praplečia kitą panaudojimo atvejo elementą.	Išplėtimo ryšys
<extensionPoint xmi:type='uml:ExtensionPoint' xmi:id='id' visibility='public' name='jei nori praplėsti'/>	Kuriamas panaudojimo atvejo elemento viduje, kuris yra praplečiamas	Išplėtimo sąlyga

Apibendrinus, 1.3.1 poskyryje buvo atlikta UML kalbos ir jos taikymo sistemoms projektuoti analizė. 1.3.1.1 poskyryje išanalizuoti UML PA diagramos pagrindiniai elementai ir taisyklės. Galiausiai 1.3.1.2 poskyryje išanalizuotos XMI generavimo galimybės PA diagramai. Toliau 1.3.2 poskyryje analizuojamos mašininio mokymosi galimybės vaizdų atpažinimui.

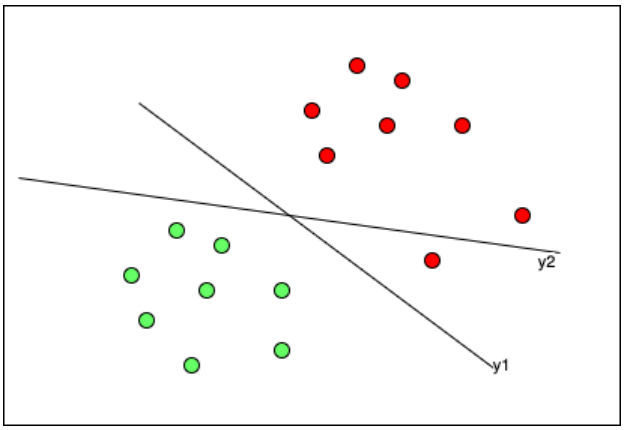
1.3.2. Mašininio mokymosi galimybės vaizdų atpažinimui

1.3.2.1. SVM

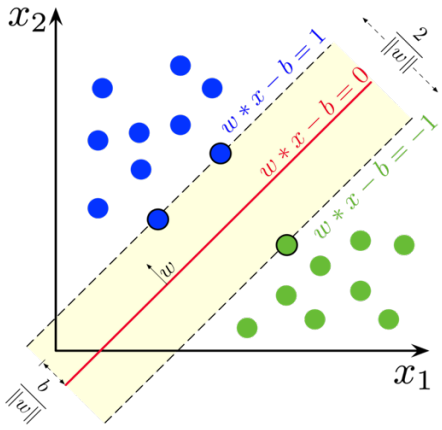
Atraminių vektorių mašina SVM (angl. *Support Vector Machine*) yra vienas iš prižiūrimo mašininio mokymosi algoritmų, naudojamas klasifikavimo ir regresijos uždaviniams spręsti. Regresijos uždavinys sprendžiamas, kai išeities rezultatas turi būti tolydi reikšmė, o klasifikavimo – diskreti. SVM priklauso generalizuotų linijinių klasifikatorių poaibiui. Šis algoritmas naudoja mašininio

mokymosi teoriją spėjimo realizavimui įgyvendinti tuo pačiu metu siekiant išvengti dažnos kitų mašininio mokymosi metodų problemos – persimokymo arba generalizacijos trūkumo [16]. Generalizacija yra modelio gebėjimas teisingai suklasifikuoti duomenis, kurie nebuvo naudojami modelio apmokymo imtyje. Yra pastebėta, kad su tam tikrais duomenų rinkiniais SVM pasiekia geresnius generalizacijos rezultatus, kai naudojama mažesnė apmokymo duomenų imtis, palyginus su neuroninių tinklų architektūromis [17].

Klasifikavimo uždaviniuose vienas iš pagrindinių uždavinių yra parinkti teisingą sprendimo ribą (angl. *decision boundary*) (1.11 pav.), kuri visus duomenis suskaidytų į teisingas sritis, priklausančias skirtingoms klasėms. Kai klasifikavimo uždavinys yra sprendžiamas tarp dviejų klasių, sprendimo riba yra tiesė, kitu atveju - hiperplokštuma. SVM klasifikatorius siekia surasti tokią sprendimo ribą, kad atstumas nuo jos iki skirtingų taškų, kertančių sprendimo ribos paraštę (angl. *margin*) ir priklausančių skirtingoms klasėms, būtų didžiausias (1.12 pav.). Taškai, kurie nulemia sprendimo ribą yra vadinami atraminiais vektoriais (angl. *support vectors*), nes tik jais vadovaujantis sudaromas sprendimo ribos vektorius.

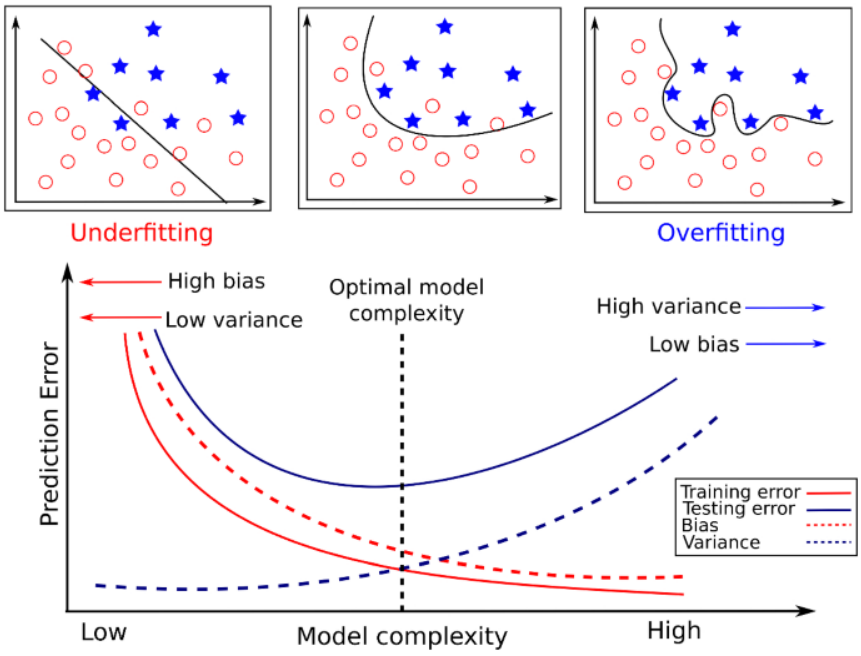


1.11 pav. Skirtingos galimos sprendimo ribos tiesės y1 ir y2



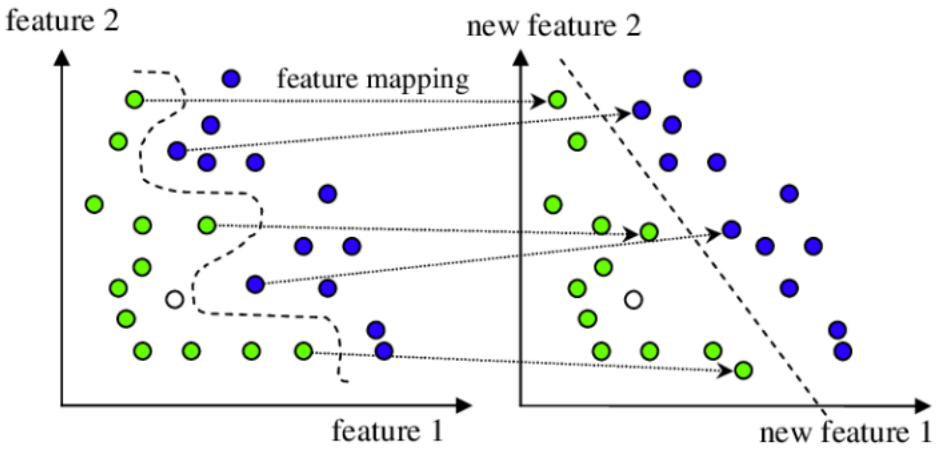
1.12 pav. Sprendimo tiesės parinkimas atraminiais vektoriais

SVM klasifikatorius yra skirstomas į minkštos paraštės klasifikatorių (angl. *Soft Margin Classifier*) ir kietos paraštės klasifikatorių (angl. *Hard Margin Classifier*) [18] [19]. Jeigu duomenys gali būti tiesiškai atskiriami, tikslinga naudoti kietos paraštės klasifikatorių, kuris parenka tokia sprendimo ribą, kad visi duomenys būtų teisingai atskirti į jiems priklausančias klasių sritis. Realiomis sąlygomis duomenys dažniausiai nebūna tiesiškai atskiriami. Verta paminėti, kad apmokymo duomenų imtys neretai pasižymi triukšmu (angl. *noise*) ir nestandartiniais nuokrypiais (angl. *outliers*) [20]. Naudojant kietos paraštės klasifikatorių kyla grėsmė parinkti tokią sprendimo ribą, kuri tiksliai atskirtų visas klases, tačiau klasifikatorius prisitaikytų prie triukšmo ir nuokrypių duomenų imtyje, kas sumažintų modelio tikslumą ir įvyktų persimokymas (1.13 pav. dešinėje viršuje) [16] [21]. Tam išvengti, naudojamas minkštos paraštės klasifikatorius, kuris leidžia modeliui turėti nedidelę paklaidą ir duomenims atsidurti netinkamoje hiperplokštumos pusėje (1.13 pav. viduryje viršuje) [22].



1.13 pav. SVM klasifikatoriaus apmokymo trūkumas (angl. *underfitting*) ir persimokymas (angl. *overfitting*) [23]

Kadangi realiomis sąlygomis duomenys dažniausiai negali būti tiesiškai padalinti į dvi klases, yra naudojamas netiesinis klasifikavimas transformacijų branduolių metodu (angl. *kernel trick*) [24]. SVM klasifikatorius yra praplečiamas įvedant transformacijos funkciją, kuri kiekvieną duomenų rinkinio tašką (vektorių) transformuoja pagal pateiktą branduolio konfigūraciją (angl. *kernel*) [25]. Transformacijos branduolių naudojimas SVM klasifikatoriuose leidžia tiesiškai neatskiriamus duomenis transformuoti į kitokį atvaizdą aukštesnėje dimensijoje, kur jie taptų tiesiškai atskiriami [24].



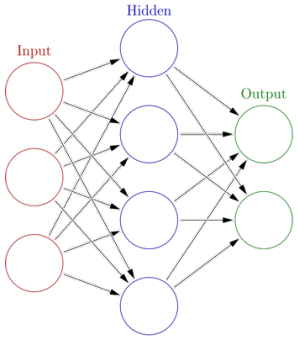
1.14 pav. Tiesiškai neatskiriami duomenys yra transformuojami į tiesiškai atskiriamus duomenis panaudojus transformacijos branduolių metodą (angl. *kernel trick*) [24]

Vaizdai pasižymi dideliu dimensijų kiekiu, tačiau pritaikius transformacijos branduolių naudojimą SVM minkštos paraštės klasifikatoriuose galima pasiekti gerų rezultatų ir sukurti tokius modelius, kurie yra labiau atsparūs persimokymui nei įprasti mašininio mokymosi modeliai [16] [21]. Verta

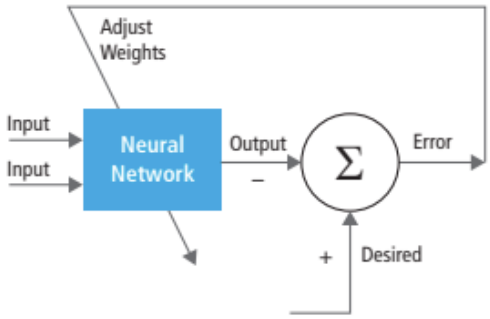
paminėti, kad SVM klasifikatoriui įprastai reikia mažesnio apmokymo duomenų rinkinio, negu giliojo mašininio mokymosi metodams [17]. Dėl šių priežasčių įvairios SVM metodų variacijos yra plačiai naudojamas vaizdų atpažinimo srityje [24] [26] [27].

1.3.2.2. CNN

Konvuliuciniai neuroniniai tinklai CNN (angl. *Convolutional Neural Networks*) yra vienas iš prižiūrimojo giliojo mašininio mokymosi algoritmų. CNN algoritmas yra pagrįstas įprastų neuroninių tinklų ANN (angl. *Artificial Neural Networks*) architektūra, todėl tikslinga ją apžvelgti, siekiant geriau suprasti CNN. ANN yra sudarytas iš dirbtinių neuronų, kurie tarpusavyje yra sujungti ryšiais - svoriais (1.15 pav.). Svoriai ir neuronai turi skaitines reikšmes, kurios yra nuolatos modifikuojamos neuroninio tinklo mokymosi eigoje atgalinio grįžimo (angl. *back propogation*) etapu (1.16 pav.). Atgalinio grįžimo etape yra paskaičiuojamas klaidos funkcijos gradientas įtraukiant prieš paskutinio sluoksnio neuronų ir svorių reikšmes. Sluoksnius jungiantys svoriai ir sluoksnyje esančių neuronų reikšmės yra modifikuojamas pagal gautą gradientą. Šis procesas yra įvykdomas visiems neuroninio tinklo sluoksniams, todėl vadinamas atgaliniu grįžimu. Verta paminėti, kad neuroninis tinklas iš esmės yra labai sudėtinga funkcija, o neuroninio tinklo apmokymas idealiu atveju yra globalaus minimumo ieškojimas, pavyzdžiui, gradientiniu metodu. Kai neuroninis tinklas nekonverguoja, sakoma, kad užstringama ties vienu iš lokalių funkcijos minimumų. ANN sluoksniai yra skirstomi į įvesties sluoksnį (angl. *input layer*), paslėptus sluoksnius (angl. *hidden layer*) ir išeities sluoksnį (angl. *output layer*) (1.15 pav.). Dirbtiniai neuroniniai tinklai, kurie turi tik vieną įeities, paslėptąjį ir išeities sluoksnį yra vadinami negilieji neuroniniai tinklai (angl. *shallow networks*). Neuroniniai tinklai (angl. *deep neural networks*) yra vadinami giliaisiais neuroniniais tinklais, kai jie turi daugiau negu vieną paslėptąjį sluoksnį. Pirmasis įvesties sluoksnis atpažįsta tam tikrus pasikartojimus (angl. *patterns*) duomenų imtyje, kurie savo ruožtu yra perduodami gilesniems paslėptiesiems sluoksniams, kur vyksta detalesnė analizė ir galiausiai prieinama prie išvesties sluoksnio, kur labiausiai suaktyvintas neuronas nusako neuroninio tinklo spėjimą duomenų įvesties vektoriui. Dirbtinis neuroninis tinklas geba atpažinti netiesinius braižus (angl. *non-linear patterns*) duomenų imtyje, nes reikšmės ateinančios į neuroną yra transformuojamos panaudojus netiesines aktyvacijos funkcijas, pavyzdžiui, sigmoidę (angl. *sigmoid*) arba *ReLU* (angl. *Rectified Linear Unit*). Skirtingi sluoksniai ar net skirtingi neuronai tame pačiame sluoksnyje gali turėti skirtingas aktyvacijos funkcijas. Neuroninių tinklų architektūros kaip CNN ir yra sudaromos kombinuojant, ir pasirenkant skirtingas neuroninių tinklų charakteristikas: aktyvacijos funkcijas, paslėptųjų sluoksnių ir neuronų juose kiekį, svorių sujungimo architektūrą, pradines neuronų reikšmes, įvesties ir išvesties sluoksnių matmenimis, tinklo apmokymo metodą, klaidos funkcijas ir kitas charakteristikas.



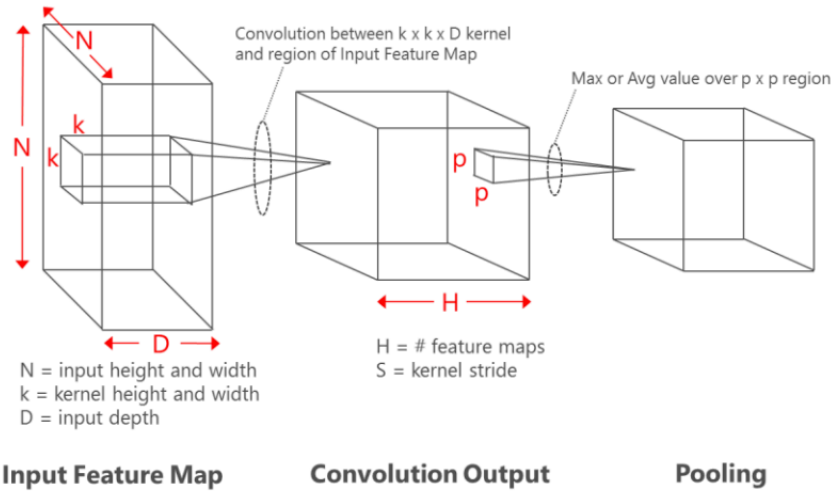
1.15 pav. Dirbtinis neuroninis tinklas [28]



1.16 pav. Neuroninio tinklo apmokymas [29]

Tipinė CNN architektūra turi nuo 5 iki 25 skirtingų paslėptųjų sluoksnių [30] [31] [32]. Kombinuojant skirtingas sluoksnių architektūras ir jas sujungiant gaunamos sudėtingos giliųjų neuroninių tinklų architektūros, kuriomis gali būti sprendžiamos klasifikavimo ar vaizdų atpažinimo problemos. CNN architektūroje dažniausiai randami šie sluoksniai: konvoliuciniai sluoksniai (angl. *convolutional layers*), telkimo/atrankos sluoksniai (angl. *pooling/subsampling layers*), netiesiniai sluoksniai (angl. *non-linear layers*) ir pilnai sujungti sluoksniai (angl. *fully connected layers*) [30] [31].

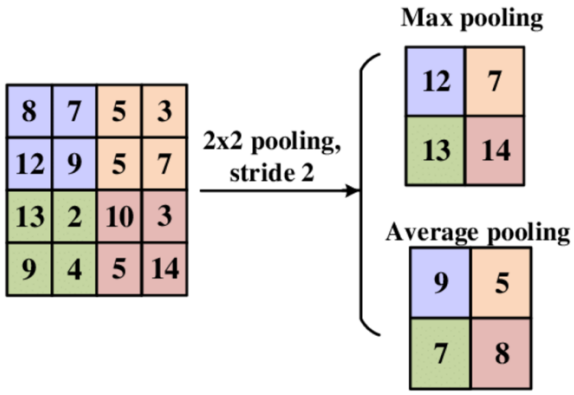
Konvoliuciniai sluoksniai yra naudojami išgauti įvairias ypatybes duomenų imtyje. Pirmieji sluoksniai padeda išgryninti žemo lygio savybes, tokias kaip kampus, linijas, kraštus. Tolimesniuose sluoksniuose išgaunamos aukštesnio lygio savybės [33]. Supaprastintas konvoliucinio sluoksnio veikimas yra pavaizduotas 1.17 pav.. Savybėms išgauti naudojamos branduolio matricos (angl. *kernels*), kurios yra *slenkamos* po vieną eilutę ir stulpelį per visą duomenų imtį ir dauginamos [33]. Gautas rezultatas yra sumažintos dimensijos duomenų imtis su išryškintomis savybėmis pagal branduolio matricą. Branduolio matricos koeficientai gali būti statiniai, pasirenkant rinkoje patikrintus branduolius, pavyzdžiui, *Sobelio* filtro koeficientus kraštų atpažinimui [34], arba išmokstami neuroninio tinklo mokymosi metu.



1.17 pav. Konvoliucinio sluoksnio veikimas [33]

CNN architektūroje telkimo ir atrankos sluoksniai sumažina duomenų imties eilutės dimensijų kiekį bei padeda apsisaugoti nuo triukšmo ir iškraipymų [35]. Duomenų imties eilutė pagal pasirinktus

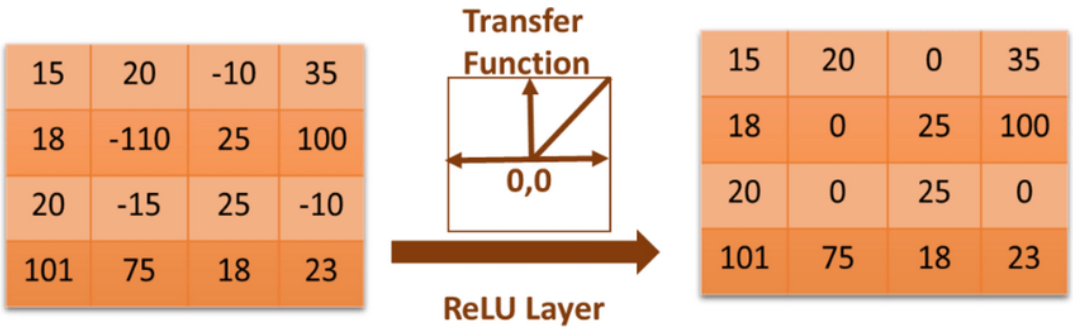
sluoksnio parametrus (regiono dydį, paslinkimo žingsnį ir t.t) yra sudalinama į nepersidengenčius regionus ir kiekvienam iš šių regionių pritaikomas maksimalaus arba vidurkio sutelkimo metodas. Maksimalaus sutelkimo metodu iš regiono išrenkamas didžiausias narys, o vidurkio sutelkimo metodu suskaičiuojamas regiono visų narių vidurkis ir apvalinamas iki sveikojo skaičiaus (1.18 pav.) [35]. Jeigu duomenų imčiai, sudarytai iš 16 stulpelių būtų pritaikytas suskaidymas į 2x2 lygius regionius, tai po sutelkimo sluoksnio duomenų imties eilutės stulpelių skaičius būtų sumažintas 4 kartus. Tai ypač svarbu, kai dirbama su didelių rezoliucijų nuotraukomis, pavyzdžiui 1000x1000, kai viena duomenų įvesties eilutė gali siekti iki 1 mln. ar net daugiau apdorojamų parametrų.



1.18 pav. Maksimalaus ir vidurkio sutelkimo metodai [35]

Netiesiniai sluoksniai CNN architektūroje yra naudojami atpažinti netiesinius braižus duomenų imtyje. Neuroninio tinklo sluoksniai yra padaromi netiesiniais, juose naudojant netiesines aktyvacijos funkcijas, pavyzdžiui *ReLU* (angl. *Rectified Linear Unit*) [36]. Dviejų dalių funkcija (angl. *piecewise function*) *ReLU* yra realizuota sąryšiu $f(x) = \max(0, x)$ ir užtikrina, kad argumento reikšmė nebūtų mažesnė už 0 (1.19 pav.) [37]. Ši aktyvacijos funkcija yra dažnai naudojama netiesinių neuroninių tinklų sluoksniuose, nes palyginus su kitomis netiesinėmis funkcijomis, pavyzdžiui, $f(x) = \tanh(x)$, *ReLU* funkcijos gradientas neurone yra lengvai apskaičiuojamas, kas reiškia jog neuroninį tinklą galima greičiau apmokinti [37]. Taip pat, šios aktyvacijos funkcijos naudojimas padeda spręsti dirbtinio intelekto srityje žinomą blėstančio gradiento (angl. *vanishing gradient*) problemą, kuri patiriama su kitomis netiesinėms funkcijoms [36]. CNN architektūros paskutiniuose sluoksniuose neretai yra naudojama pilnai sujungtų sluoksnių architektūra, kuri pateikia galutinius rezultatus matematiškai susumuodama praeitų sluoksnių gautus įverčius [30] [31]. Labiausiai suaktyvintas

neuronas paskutiniame sluoksnyje nusako spėjamą klasę, o bendra visų klasių panašumo rodiklių suma turi būti lygi 1.



1.19 pav. ReLU aktyvacijos funkcijos veikimas [37]

CNN architektūrose naudojami konvoliucijos ir sutelkties sluoksniai padeda apsisaugoti nuo triukšmo ir nuokrypių duomenų imtyje [35]. Taip pat, CNN architektūros pasižymi greitesniu apmokymu ir mažesniu atminties sunaudojimu įrenginiuose palyginus su įprastomis pilnai sujungtomis ANN architektūromis [33] [35]. Taigi, CNN pagrindu sukurtos architektūros yra sėkmingai naudojamos vaizdų atpažinimo problematikoje. *LeNet-5* giliojo neuroninio tinklo modelis pasiekė 98% procentų tikslumą *Fashion MNIST* duomenų rinkiniui ir pralenkė kitus iki tol rinkoje sukurtus nuotraukų klasifikavimo metodus [38]. *NFNet-F4+* CNN pagrindu sukurta architektūra pasiekė 89.2% tikslumą *ImageNet* duomenų rinkiniui su 1000 skirtingų klasių ir pasiekė vieną geriausių tuo metu rezultatų [39].

1.3.3. Teksto atpažinimo galimybės

1.3.3.1. OCR apžvalga

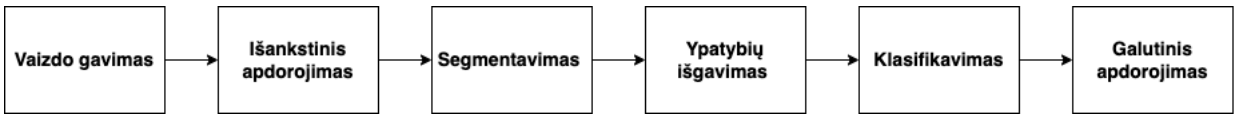
Optinis simbolių atpažinimas OCR (angl. *Optical Character Recognition*) yra procesas, kurio metu iš įvairių šaltinių kaip ranka rašytų dokumentų, atspausdintų tekstų, nuotraukų yra išgaunamas tekstas ir paverčiamas į skaitmeninį formatą [40]. OCR yra laikomas vienas iš skaitmeninių nuotraukų apdorojimo DIP (angl. *Digital Image Processing*) srities taikymų [41]. OCR sąsajos gali būti grupuojamos į 3 pagrindines kategorijas [42]:

1. OCR sąsajos, kurios yra pritaikytos veikti tik ranka rašytiems tekstams (angl. *handwritten*).
2. OCR sąsajos, kurios yra pritaikytos veikti tik skaitmeniniu būdu parašytiems (angl. *machine printed*) tekstams.
3. OCR sąsajos, kurios yra pritaikytos prie konkrečių sąlygų ir situacijų: kalbos, šrifto, specifinių simbolių.

Verta paminėti, kad nors OCR šiomis dienomis yra dažniausiai naudojamas teksto atpažinimui ir skaitmenizavimui, OCR buvo sukurtas XX a. pradžioje siekiant padėti žmonėms, turintiems regėjimo sutrikimų, naudotis rašytiniais tekstais perskaitant juos garsiai mašinų pagalba [42] [43]. Dėl technologijų išsivystymo ir skaičiavimo galios trūkumų OCR technologija iki pat XX a. vidurio nebuvo plačiai prieinama ir tobulinama, nes buvo susiduriama su įvairiais greičio ir tikslumo sunkumais. Pirmosios ir tuo metu geriausios OCR sąsajos gebėdavo atpažinti tik po 1 žodį per minutę [40].

1.3.3.2. OCR veikimas

OCR sąsajų veikimas susideda iš daug skirtingų etapų, tačiau literatūroje OCR veikimas dažniausiai yra suskaidomas į 6 pagrindinius etapus (1.20 pav.): vaizdo gavimą (angl. *image acquisition*), išankstinį apdorojimą (angl. *pre-processing*), segmentavimą, ypatybių išgavimą (angl. *feature extraction*), klasifikavimą, galutinį apdorojimą (angl. *post-processing*) [42] [44].

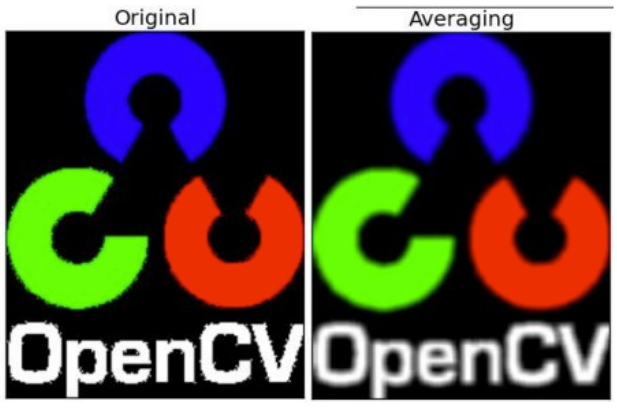


1.20 pav. Pagrindiniai OCR sąsajų etapai [42]

Vaizdo gavimo etapas yra pirmasis ir vienas svarbiausių ne tik OCR sąsajų veikime, bet ir DIP srityje, nes be išgauto vaizdo nebūtų ką apdoroti. Šiame etape yra išgaunami vaizdai iš kamerų, sensorių arba gali būti skaitmenizuojami. Šiame etape nuotrauka dažniausiai būna visiškai neapdorota (angl. *raw input*), todėl toliau yra perduodama į išankstinio apdorojimo etapą.

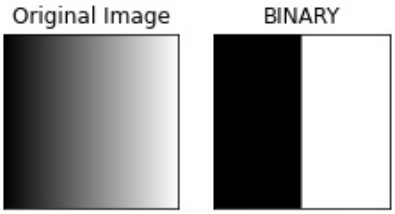
Išankstinio apdorojimo etapas, susidedantis iš įvarių operacijų skaitmenizuotam vaizdui yra kritinis norint pasiekti aukštą tikslumą OCR sąsajose [42]. Šis etapas dažnai pradedamas vaizdo konvertavimu iš daug atspalvių sudaryto vaizdo į dviejų atspalvių, juodo ir balto, vaizdo konvertavimą (angl. *binarization*). Toliau gali būti naudojamos įvairios technikos siekiant pagerinti nuotraukų detalumą ir kokybę [45]:

– Filtravimo operacijos (angl. *filtering operations*) – naudojamos įvairios branduolio matricos operacijos su skaitmenizuotų nuotraukų pikseliais, kurios galėtų išryškinti nuotraukoje esančių objektų kraštus [34] arba išlyginti nuotrauką (angl. *image smoothing*) sumažinant triukšmą.



1.21 pav. Originali nuotrauka ir nuotrauka po išlyginimo operacijos [46]

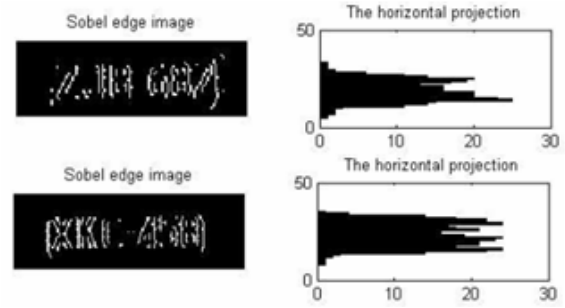
– Slenksčio pritaikymas (angl. *tresholding*) – operacija, kuri dažniausiai atliekama binarizuotoms nuotraukoms (angl. *gray scale*) ir kuria siekiama sumažinti triukšmą ir išryškinti esmines nuotraukų sritis bei izoliuoti tas sritis nuotraukuose, kurios yra nereikšmingos. Slenksčio taikymas gali būti globalus, kai viena slenksčio reikšmė yra taikoma visai nuotraukai, arba adaptyvus, kai skirtingiems pikseliams taikomos skirtingos slenksčio reikšmės.



1.22 pav. Originali nuotrauka ir nuotrauka, kuriai buvo pritaikyta slenksčio operacija [47]

– Vaizdo pasukimo koregavimas (angl. *image skew correction*) – vaizdai gali būti kaip nors pasukti ar iškraipyti, todėl tai apsunkina OCR sąsajų atpažinimo tikslumą [48]. Tokie duomenys turi būti koreguojami prieš pateikiant juos tolimesniems etapams. Pavyzdžiui, nuotraukų pasukimų pataisymams gali būti naudojamos *Hough* transformacijos.

Atlikus vaizdo gavimo ir išankstinio apdorojimo etapus yra vykdomas nuotraukos segmentavimas. Tai procesas, kurio metu nuotrauka yra padalinama į atskirus regionus, kurie izoliuotai gali būti apdorojami tolimesniuose etapuose. OCR sąsajose nuotrauka yra susegmentuojama į atskirus regionus pagal teksto eilutes (angl. *line level segmentation*) ir žodžius (angl. *word level segmentation*) [49]. Jeigu apdorojama nuotrauka su ranka rašytu tekstu, gali reikėti atlikti papildomą segmentavimą pagal simbolius (angl. *character level segmentation*). Segmentavimui dažniausiai yra naudojamas histogramos projekcijos metodas HPM (angl. *Histogram Projection Method*) [49]. Binarizuotos nuotraukos pikseliai gali būti sugrupuojami į dvi kategorijas: priekinio plano pikselius (angl. *foreground pixels*) ir antrinio plano pikselius (angl. *background pixels*). Priekinio plano pikseliai nuotraukose yra svarbiausi, nes savyje laiko naudingą informaciją. Pikselis priskiriamas į vieną iš šių kategorijų nuotraukos binarizavimo etapu. HPM metodu yra suskaičiuojama, kiek priekinio plano pikselių yra nuotraukoje pagal pasirinktą ašį [50]. Pritaikius HPM metodą pagal horizontalią ašį, būtų gaunama histograma, kuri nusakytų kiek priekinio plano pikselių yra kiekvienoje nuotraukos eilutėje (1.23 pav.).

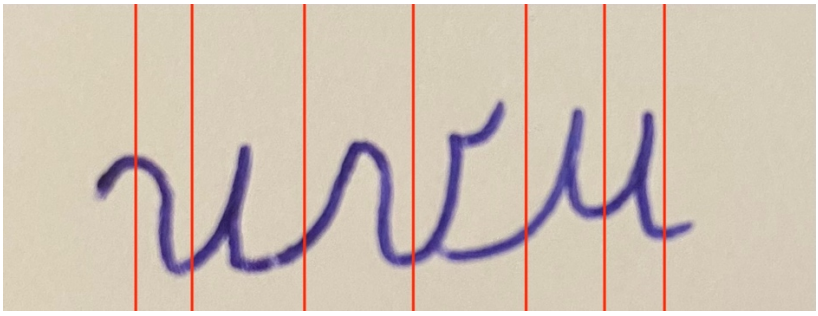


1.23 pav. HPM metodas pritaikytas nuotraukai pagal horizontalią ašį [50]

Nuotrauka gali būti segmentuojama eilutėmis pritaikius HPM metodą pagal horizontalią ašį [49]. Gautos horizontalios projekcijos histogramos eilutės, kurios neturi priekinio plano pikselių arba jų turi labai nedaug indikuoja, kad tai yra tuščios eilutės. Tuščia eilutė yra naudojama nuotraukos segmentavimui. Toliau izoliuotai teksto eilutei yra pritaikomas HPM metodas pagal vertikalią ašį, kas leidžia susegmentuoti teksto eilutę į žodžius [49]. Įprastai anglų kalboje tarpai tarp raidžių viename žodyje būna mažesni, negu tarpai tarp atskirų žodžių. Tai žinant, iš gautos vertikalios projekcijos histogramos galima pastebėti, kad stulpelių neturinčių aktyvių pikselių besitęsiantis

skaičius yra didesnis tarp žodžių tarpų, negu tarp raidžių tarpų viename žodyje. Turint tokią informaciją eilutę galima sėkmingai susegmentuoti po vieną žodį.

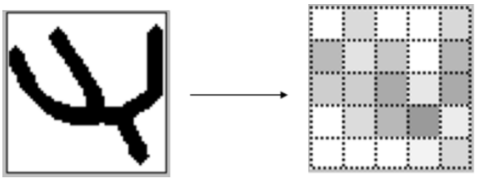
Tolimesnis žodžio segmentavimas į simbolius priklauso nuo duomenų su kuriais yra dirbama. Jeigu OCR sąsajai yra pateikiamas tekstas, kur simboliai yra atskirti tarpeliais, segmentavimas į simbolius gali būti atliekamas panaudojus tokią pačią metodiką, kaip ir segmentuojant žodžius. Kitu atveju, jeigu pateikiamas ranka kursyvu rašytas tekstas (angl. *cursive handwritting*) žodžių segmentavimas į simbolius tampa itin sudėtinga užduotimi, nes raidės tarpusavyje yra sujungtos. Pavyzdžiui, raidei *u* arba *v* atlikus vertikalią projekciją, būtų tokia vieta projekcijos stulpelyje, kur yra tik vienas aktyvus pikselis kaip ir būna su sujungimais tarp raidžių. Tampa sudėtinga nustatyti ar reikėtų simbolį atskirti ties raidės *u* viduriu, ar ties sujungimu tarp raidžių. Ši problema OCR sąsajų srityje yra geriau žinoma kaip per didelio segmentavimo problema (angl. *over segmentation problem*) [51].



1.24 pav. Galimi simbolių atskyrimo variantai

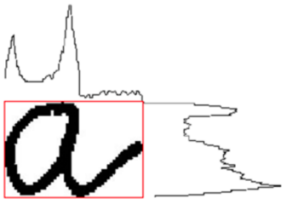
Atlikus nuotraukos segmentavimą yra vykdomas ypatybių išgavimo etapas, kurio rezultatai yra naudojami klasifikavimo etape. Kiekviena raidė yra reprezentuojama vektoriumi, kuris yra sudaromas pritaikius įvairius statistinius ir struktūrinius ypatybių išgavimo metodus [52]:

- Zonavimas (angl. *zoning*) – visi raidės segmento pikseliai yra padalinami į lygias dalis, pavyzdžiui 2x2, 4x4. Padalintuose regionuose paskaičiuojama, kiek yra priekinio plano pikselių (angl. *foreground pixels*). Gautas rezultatas apibrėžia vieną iš raidės vektoriaus ypatybių [52].



1.25 pav. Zonavimo technika [52]

- Histogramos projekcijos metodas (angl. *Projection Histograms*) – skaičiuojamas pikselių skaičius stulpeliuose ir eilutėse [52]. Šis metodas gali padėti tiksliai atskirti raides *m* ir *n*.



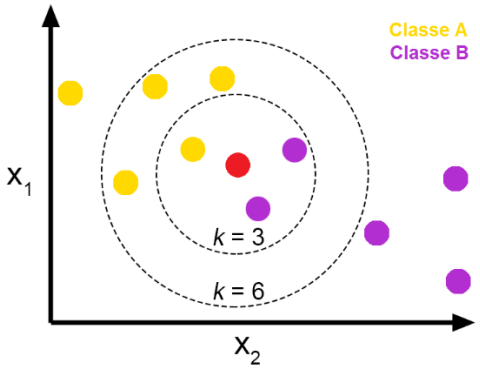
1.26 pav. Histogramos projekcija raidei *a* [52]

– Atstumų profiliai (angl. *distance profile*) – skaičiuojamas atstumas pikseliais nuo nuotraukos rėmų iki raidės kraštų. Atstumų profiliai gerai apibrėžia raidžių struktūrą ir leidžia atskirti didžiąją dalį raidžių, pavyzdžiui *p* ir *q*.

1.27 pav. Atstumų profilis raidei *a* [52]

Sudarius simbolių ypatybių vektorius yra vykdomas simbolių klasifikavimo etapas. Klasifikavimo etapui realizuoti yra naudojamos įvairios metodikos, tačiau literatūroje dažniausiai galima pastebėti naudojamus tris klasifikavimo metodus: KNN, SVM bei ANN [42] [52] [53]:

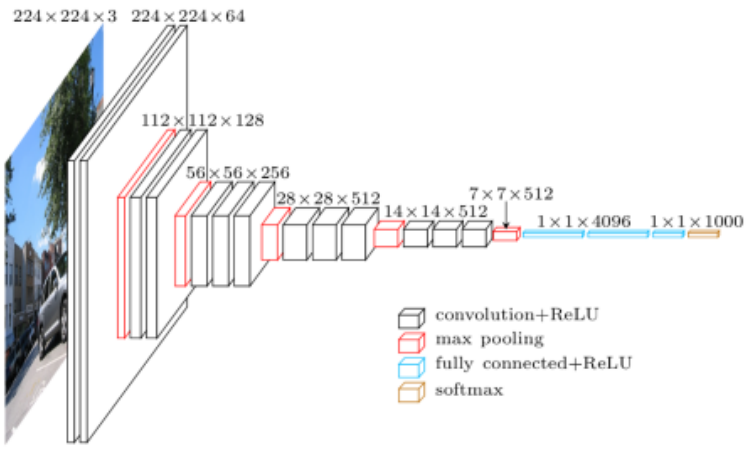
– K – arčiausių kaimynų metodas KNN (angl. *K-nearest neighbour technique*) yra prižiūrimojo mokymo klasifikavimo algoritmas. KNN algoritmas duomenų imties eilutei suranda nurodytą skaičių K arčiausių kaimynų pagal euklido atstumą (angl. *euclidean distance*). Toliau yra vykdomas balsavimo etapas, kur pagal tai, kokios klasės arčiausių kaimynų turi duomenų eilutė pastaroji yra priskiriama tai klasei. Tokiu būdu simboliai yra suklasifikuojami į atitinkamas klases.



1.28 pav. KNN klasifikavimo pavyzdys [54]

– SVM (1.3.2.1) atraminių vektorių metodas yra prižiūrimojo mokymosi algoritmas. Šiuo metodu duomenų eilutės vektoriui yra bandoma surasti teisingą sprendimo ribą, kuri priskirtų simbolį į teisingą klasę. Priešingai nei KNN, šiam metodui reikia numatyti ir parinkti daugiau parametrų kaip paraštės tipą bei optimalią branduolio matricą.

– Naudojamos įvairios dirbtinių neuroninių tinklų architektūros, pavyzdžiui CNN (1.3.2.2) gilieji neuroniniai tinklai [55]. *Telugu* kalbos simboliams panaudojus vieną iš CNN architektūros modelių VGG-16 su 1600 simbolių apmokymo imtyje buvo pasiektas 92% tikslumas [55].



1.29. pav. VGG-16 CNN architektūra [56]

Galutinio apdorojimo etape yra apjungiami klasifikavimo rezultatai ir vykdomas gautų rezultatų pataisymas. Vienas iš būdų kaip pagerinti OCR sąsajos rezultatus yra lyginti juos su atitinkamos kalbos žodynais ir atlikti pataisymus [44]. Šiame etape gali būti naudojami ir įvairūs mašininio modelio metodai – SVM, ANN - kurie padėtų nustatyti nekorektiškus rezultatus pagal įvairias charakteristikas, kaip konkrečių simbolių skaičių, dažnumą, žodžio ilgumą [57].

1.3.3.3. OCR sąsajų palyginimas

OCR sąsajų analizės metu buvo apžvelgti 3 rinkoje plačiai naudojami OCR sprendimai: *Tesseract*, *Amazon Textract, Google Cloud Vision*. *Tesseract OCR* yra atviro kodo biblioteka sukurta *Google*. *Tesseract OCR* gali būti pritaikytas konkrečiam sprendimui ir integruotas į įvairias programas ir platformas. Nors ši biblioteka yra nemokama ir konfigūruojama aukštą tikslumą galima pasiekti tik naudojant geros kokybės eskizą ir atlikus paveikslėlio apdorojimo žingsnius, kurie apsunkina prototipo įgyvendinimą. *Amazon Textract* ir *Cloud vision* yra atitinkamai *Amazon Web Services* ir *Google* teikiamos OCR debesijos paslaugos. Abu šie sprendimai skirti tekstui ir duomenims išgauti iš nuskenuotų dokumentų, PDF failų ir vaizdų. Pasitelkdami mašininio mokymosi algoritmus šie sprendimai nustato ir dokumentuose randa tekstą, lenteles ir formas. Papildomai gaunamas atpažintų teksto elementų ribojantis langelis su koordinatėmis, todėl tai galėtų supaprastinti PA diagramos simbolių atpažinimo modulį, nes nereikėtų diagramos simbolio atpažinimo modeliui atskirti teksto klasės. Svarbu paminėti, kad priešingai nei atvirojo kodo *Tesseract* sprendimas, *Amazon Textract* ir *Google Cloud Vision* paslaugos yra mokamos priklausomai nuo naudojimo intensyvumo.

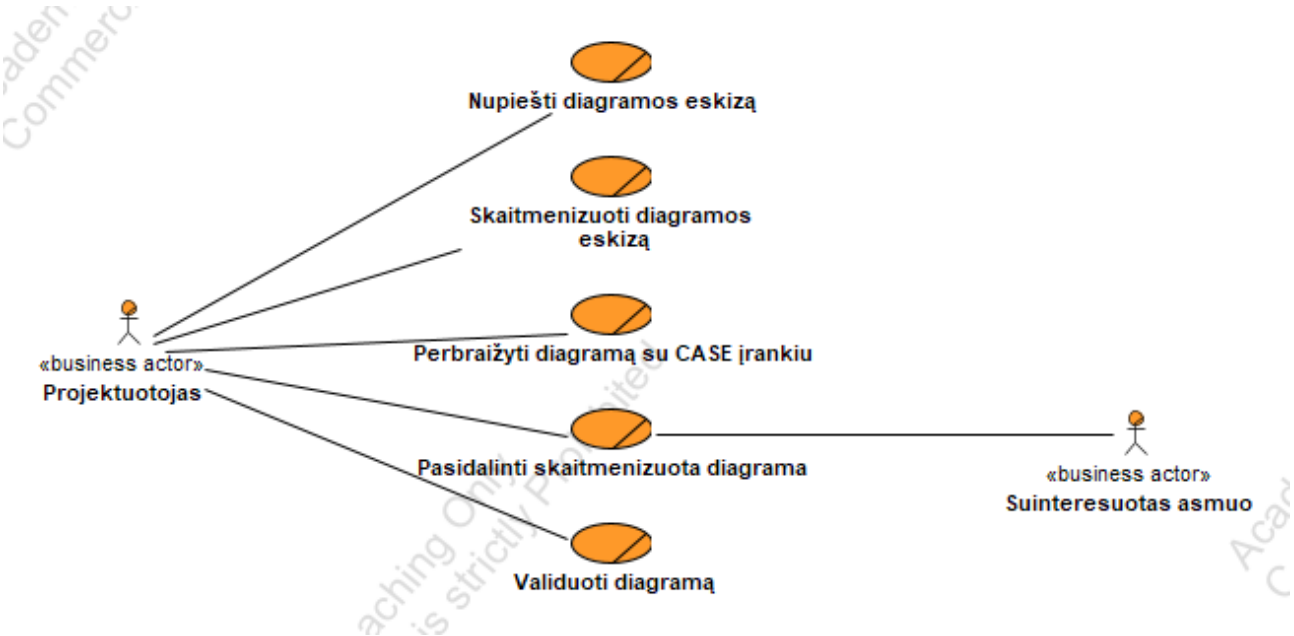
1.3 lentelė. Tesseract, Amazon Textract ir Google Cloud Vision sprendimų palyginimas

Kriterijus	**Tesseract [58]**	**Amazon Textract [59]**	**Google Cloud Vision [60]**
Šaltinis	Atvirojo kodo	AWS debesijos teikiama paslauga	Google Cloud debesijos OCR paslauga

Pritaikymas	Plačiai konfigūruojamas konkretiems panaudojimo atvejams	Riboto konfigūravimo	Riboto konfigūravimo
Tikslumas [61]	~80%	~95%	~95%
Integravimo galimybės	Reikalinga įdiegti rankiniu būdu į vykdomąją aplinką	Galima naudoti per HTTP protokolą	Galima naudoti per HTTP arba gRPC protokolą
Randami teksto regionai	Ne	Taip	Taip
Kaina	Nemokamai	Mokama pagal išnaudotų API kvietimų kiekį	Mokama pagal išnaudotų API kvietimų kiekį
Dokumentų formatai	Įvairūs paveikslėlio formatai	Įvairūs paveikslėlio formatai, PDF failai, dokumentai	Įvairūs paveikslėlio formatai, PDF failai, dokumentai
OCR technologija	Naudojami mašininio mokymosi algoritmai	Naudojami mašininio mokymosi algoritmai	Naudojami mašininio mokymosi algoritmai

Taigi, norint naudoti *Tesseract OCR* ir pasiekti aukštą tikslumą reikia atlikti papildomą paveikslėlio apdorojimą [58], kas apsunkintų OCR komponento sukūrimą. Taip pat, norint integruoti *Tesseract OCR* reikalinga įdiegti įrankį į vykdomąją aplinką rankiniu būdu. Kita vertus, *Amazon Textract* ir *Google Cloud Vision* teikiamos OCR debesijos paslaugos pasižymi aukštesniu tikslumu be papildomo paveikslėlio apdorojimo ir platesnėmis integravimo galimybėmis, nes suteikiama galimybė komunikuoti su sąsajomis per HTTP arba gRPC protokolus. Kita svarbi savybė *Amazon Textract* ir *Google Cloud Vision* sąsajų yra tai, kad šios sąsajos gražina teksto regionus eskize, todėl diagramos simbolių atpažinimo komponentui nereikėtų gebėti aptikti teksto klasės.

1.4. Tyrimo objekto naudotojų analizė



1.30 pav. Tyrimo objekto naudotojų veiklos analizė PA atvejais

Nors šiuo metu rinkoje egzistuoja daug įrankių, leidžiančių braižyti skaitmenizuotas UML diagramas, praktikoje dažnai pirmuosius diagramų eskizus projektuotojai nubraižo ranka ant popieriaus arba išmaniosios lentos (angl. *white board*) diskusijų metu tarp sistemos kūrėjų ir užsakovo [2]. Norint

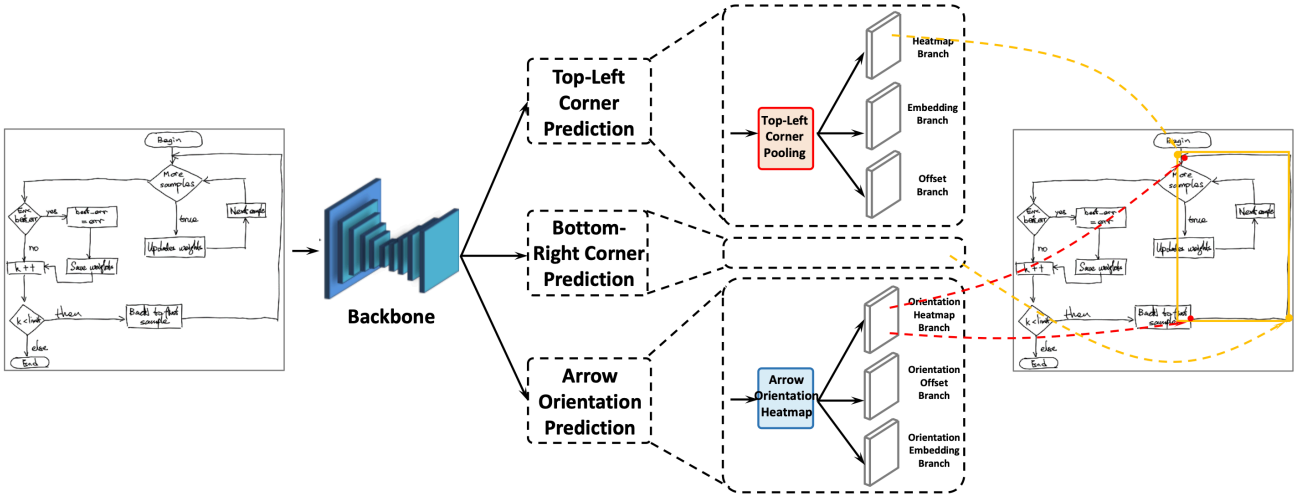
efektyviai pasidalinti eskizu su kitais interesuotais asmenimis bei toliau plėtoti pradinį diagramos eskizą, pastarąjį reikia rankiniu būdu perbraižyti su CASE įrankiu. Šiame žingsnyje galima patobulinti procesą automatiškai skaitmenizuojant diagramos eskizą.

1.5. Esamų problemos sprendimų analizė

Ranka braižytų diagramų atpažinimo metodikos literatūroje skirstomas į dvi pagrindines kategorijas: realaus laiko braižomų diagramų (angl. *online-targeted*) ir eskizų (angl. *offline-targeted)* atpažinimą ir skaitmenizavimą [3]. Skirtingoms kategorijoms yra taikomi skirtingi sprendimo būdai. Realaus laiko braižomų diagramų atpažinimui (angl. *online-targeted*), kai turimi istoriniai duomenys laike ir erdvėje, naudojami įvairūs skaitiniai metodai bei sprendžiami optimizavimo uždaviniai, kuriais siekiama atpažinti ir sugrupuoti UML simbolius. Diagramų atpažinimui iš eskizų taikomi vaizdų atpažinimo ir mašininio mokymosi algoritmai, kurie gali apdoroti bendresnius atvejus. Verta paminėti, kad diagramų atpažinimo metodai iš eskizų gali būti pritaikomi realaus laiko braižomoms diagramoms, pavyzdžiui, diagramoms ant išmaniųjų lentų (angl. *white-board*) atpažinti. Pastarosios kategorijos sprendimo būdai, diagramoms iš ranka braižytų eskizų atpažinimui ne visada gali būti sėkmingai panaudoti dėl tokių metodų priklausomybės nuo istorinių braižymo duomenų, kurie ranka braižytuose eskizuose neegzistuoja.

1.5.1. DrawNet gilusis neuroninis tinklas

Šio darbo kontekste nagrinėjamos egzistuojančios sąsajos diagramų atpažinimui iš eskizų (angl. *offline-targeted*). Dalis egzistuojančių mašininio mokymosi modelių ir vaizdų atpažinimo algoritmų gali sėkmingai atpažinti pavienius ranka pieštų diagramų simbolius kaip apskritimus, linijas, rodykles, kvadratus, tačiau negeba efektyviai apjungti atpažintų simbolių į vientisą diagramos struktūrą. Kita dalis metodų remiasi istoriniais diagramos eskizo piešimo duomenimis, kas užkerta kelią šių sąsajų praktiškumui ir diagramų skaitmenizavimui iš eskizų. CNN neuroninių tinklų architektūromis grįsti neuroniniai tinklai šiuo metu vis dar plačiai naudojami dirbant vaizdų atpažinimo problematikoje. Viena iš tokių egzistuojančių sąsajų yra *DrawNet* gilusis neuroninis tinklas, pagrįstas CNN raktinių taškų (angl. *keypoints*) aptikimu, gebančiu sėkmingai atpažinti pavienius diagramų elementus [3]. *DrawNet* neuroninis tinklas yra sukurtas *CornerNet* tinklo pagrindu papildant jį dviem naujais raktinių taškų nustatymo moduliais ir rodyklių orientacijos nustatymo moduliu. *DrawNet* gilusis neuroninis tinklas kaip bazinę architektūrą (angl. *backbone*) naudoja smėlio laikrodžio (angl. *hourglass*) struktūrą, kuri yra dažna kodavimo ir dekodavimo (angl. *encode-decode*) struktūra plačiai naudojama objektų raktinių taškų aptikimo uždaviniuose, pavyzdžiui, nustatant objektų poziciją. Šią struktūrą papildo viršutinio kairiojo kampo aptikimo, apatinio dešiniojo kampo aptikimo ir rodyklių orientacijos nustatymo moduliai. *DrawNet* neuronineme tinkle kiekvienas diagramos simbolis yra aptinkamas ir apibūdinamas kaip pora raktinių taškų, kurie yra viršutinis kairysis kampas ir apatinis dešinysis kampas. Ši taškų pora nusako ribojantį langą (angl. *bounding box*) diagramos simboliui. Papildomai šis neuroninis tinklas aptinka diagramoje esančių rodyklių raktinius taškus, kas leidžia nustatyti jų orientaciją. Bazinė tinklo struktūra (angl. *backbone*) kartu su lygiagrečiai veikiančiais trimis moduliais leidžia ne tik sėkmingai aptikti diagramos simbolius bet ir juos sujungti taip sukuriant pilną diagramos struktūrą.



1.31 pav. Kodavimo – dekodavimo neuroninio tinklo architektūra veikianti kartu su trimis papildomais, viršutinio kairiojo kampo atpažinimo, apatinio dešiniojo kampo atpažinimo ir rodyklės orientacijos nustatymo, moduliais [3].

DrawNet neuroninis tinklas buvo apmokintas su trimis skirtingais duomenų rinkiniais (*FC_A, FC_B, FA*). *FC_A* ir *FC_B* duomenų rinkinyje buvo pateiktos srauto (angl. *flowchart*) diagramos, o *FA* duomenų rinkinyje – baigtinių būsenų (angl. *finite automata*) diagramos. *FC_A* duomenų rinkiniui buvo naudojamos 248 diagramos modelio apmokymui ir 171 testavimui; *FC_B* rinkiniui – 280 apmokymui, 196 testavimui ir 196 validavimui. Baigtinių būsenų duomenų rinkinys FA, buvo išskaidytas į 132 diagramas apmokymui, 84 testavimui ir 84 validavimui.

1.4 lentelė. Naudojami duomenų rinkiniai *DrawNet* neuroniniam tinklui apmokinti ir įvertinti

Duomenų rinkinys/Diagramų Išskaidymas	**Bendra suma**	**Apmokymo imtis**	**Testavimo imtis**	**Validavimo imtis**
FC_A	419	248	171	-
FC_B	672	280	196	196
FA	300	132	84	84

Bendras tikslumas buvo matuojamas pagal teisingai atpažintų diagramų skaičių testavimo imtyje. Teisingai atpažinta diagrama yra laikoma tokia, kai visi diagramos simboliai yra atpažinti ir sujungti teisingai. *FC_A* duomenų rinkiniui buvo pasiektas 70.8% tikslumas; *FC_B* rinkiniui – 80.9%; *FA* – 85%. Taip pat, galima įvertinti ir pavienių klasių atpažinimo tikslumą, kuris yra ženkliai aukštesnis negu bendras diagramų atpažinimo tikslumas. Tai parodo, kad sunkiausia užduotis diagramų skaitmenizavimo iš eskizų problematikoje yra ne pavienių diagramos simbolių atpažinimas, tačiau gebėjimas juos sujungti į galutinę diagramos struktūrą.

1.5 lentelė. Srauto diagramų simbolių atpažinimo tikslumas *FC_A* ir *FC_B* duomenų rinkiniams, testavimo imčiai.

Klasė	**Tikslumas (FC_A)**	**Tikslumas (FC_B)**
Rodyklė (angl. *Arrow*)	95.7%	98.6%
Jungtis (angl. *Connection*)	99.6%	100%

Duomenys (angl. *Data*)	99.9%	100%
Sprendimo taškas (angl. *Decision*)	100%	100%
Procesas (angl. *Process*)	100%	96.4%
Baigimo taškas (angl. *Terminator*)	100%	100%
Tekstas	98.3%	99.5%
Bendras tikslumas	98.4%	99%

1.6 lentelė. Baigtinių būsenų diagramų simbolių atpažinimo tikslumas FA duomenų rinkiniui, testavimo imčiai.

Klasė	**Tikslumas**
Rodyklė (angl. *Arrow*)	98.6%
Galutinė būsena (angl. *Final state*)	100%
Būsena (angl. *State*)	100%
Tekstas	99.6%
Bendras tikslumas	99.5%

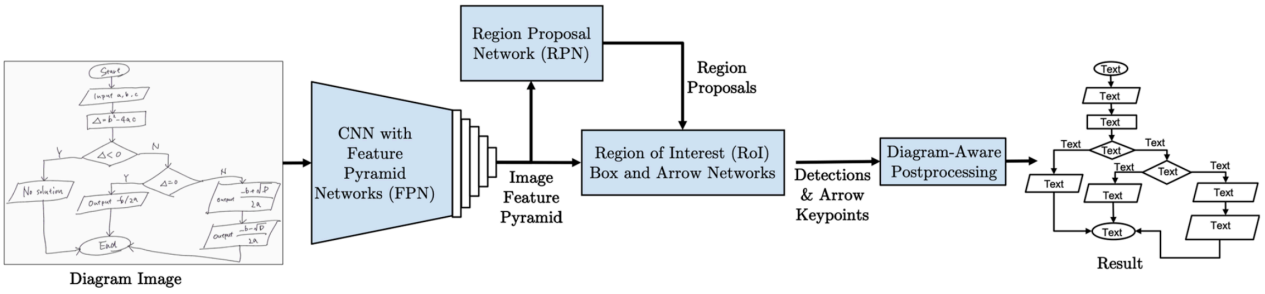
1.5.2. Arrow R-CNN gilusis neuroninis tinklas

Arrow R-CNN [62] gilusis neuroninis tinklas yra pagrįstas *Faster R-CNN* neuroniniu tinklu, kuris savo ruožtu yra kilęs iš *R-CNN* tinklo. Norint geriau suprasti *Arrow R-CNN* veikimą ir galimybes tikslingą apžvelgti pirmtako *Faster R-CNN* neuroninio tinklo architektūrą, kuri yra sudaryti iš trijų pagrindinių sluoksnių:

1. CNN ypatybių piramidžių neuroninio tinklo (angl. *CNN with Feature Pyramid Networks FPN*);
2. Objektų regionų siūlymo tinklo (angl. *Regional Proposal Network RPN)*;
3. Dominančio regiono (angl. *Region of Interest*) rėmelio pasiūlymo tinklo (angl. *RoI box network*).

CNN bazinis tinklas yra naudojamas išgauti vaizdo ypatybių vektorius, kurie naudojami RPN tinkle siekiant sudaryti didelę RoI aibę. RoI rėmelio pasiūlymo tinkle nustatoma objekto klasė ir sudaromas objekto ribojantis rėmelis. *Faster R-CNN* tinklas geba tiksliai atpažinti pavienius diagramos objektus, tačiau neįvertina bendros diagramos struktūros. *Arrow R-CNN* neuroninis tinklas praplečia *Faster R-CNN* tinklo galimybes ir sprendžia šią problemą pridedant du papildomus modulius:

1. Rodyklės pradžios ir pabaigos raktinių taškų aptikimo modulį, veikianti lygiagrečiai su RoI rėmelio siūlymo moduliu;
2. Galutinio apdorojo modulį, kuris naudoja aptiktus diagramos simbolius ir raktinius rodyklių taškus sudarant galutinę diagramos struktūrą.



1.32 pav. *Arrow R-CNN* pagrindinis veikimo principas [62]

Arrow R-CNN veikimą galima išskaidyti į tokius pagrindinius žingsnius: CNN bazinis tinklas (ResNet-101) iš pateikto vaizdo sudaro požymių vektorius. RPN tinklas pagal požymių vektorius sudaro RoI aibę. RoI rėmelio siūlymo tinklas klasifikuoja objektą ir pasiūlo rėmelį. Papildomai aptiktoms rodyklėms sudaromi galvos (angl. *head*) ir uodegos (angl. *tail*) raktiniai taškai. Galutinio apdorojimo modelis sujungia aptiktus elementus į bendrą struktūrą.

Siekiant pagerinti *Arrow R-CNN* modelio tikslumą ir prisitaikymą prie didelės diagramų variacijos naudojami duomenų praplėtimo (angl. *data augmentation*) metodai. Taip pat, buvo pastebėta, kad modelis gali sumaišyti tekstą su rodyklėmis, todėl atlikus duomenų praplėtimo žingsnius buvo padidintas atpažinimo tikslumas teksto ir rodyklių klasėms. Apmokant *Arrow R-CNN* modelį naudojami tokie duomenų praplėtimo metodai:

1. Didelės rezoliucijos nuotraukų sumažinimas išlaikant kraštinių santykį;
2. Diagramų praplėtimas iki 3 atsitiktinių žodžių;
3. Nuotraukų paslinkimas rėžyje [-0.01, 0.01], mastelio keitimas [-0.2, 0.0] ir nuotraukų pasukimas $[-5^{\circ}, 5^{\circ}]$;
4. Atsitiktinių nuotraukų pasukimas 90 laipsnių vieną arba keletą kartų;
5. Vaizdo apvertimas horizontaliai, vertikaliai arba abejomis kryptimis.

Kaip ir *DrawNet* neuroninio tinklo apmokymo atveju, naudojami *FC_A*, *FC_B*, *FA* duomenų rinkiniai. *Arrow R-CNN* tinklas papildomai buvo apmokintas su *DIDI* srauto diagramų duomenų rinkiniu, kurioje egzistuoja 22,287 diagramos su tekstinėmis anotacijomis ir 36,368 diagramos be tekstinių anotacijų. Bendras modelio tikslumas matuojamas pagal teisingai atpažintų diagramų skaičių testavimo imtyje. Teisingai atpažinta diagrama yra laikoma tokia, kai visi diagramos simboliai yra atpažinti ir sujungti teisingai. *Arrow R-CNN* modelis turi aukštą pavienių diagramos klasių atpažinimo tikslumą, o bendras teisingai atpažintų diagramų tikslumas per visus duomenų rinkinius siekia 68%.

1.7 lentelė. *Arrow R-CNN* modelio simbolių atpažinimo tikslumas *FC_A, FC_B* ir *DIDI* duomenų rinkiniams.

Klasė	**Tikslumas (FC_A)**	**Tikslumas (FC_B)**	**Tikslumas(DIDI su tekstinėmis anotacijomis)**	**Tikslumas(DIDI be tekstinių anotacijų)**
Rodyklė (angl. *Arrow*)	95.7%	98.6%	99.2%	97.5%
Jungtis (angl. *Connection*)	99.6%	100%	-	-
Dėžė (angl. *Box*)	-	-	99.9%	96.5%
Duomenys (angl. *Data*)	99.9%	100%	-	-
Rombas (angl. *Diamond*)	-	-	99.9%	97.5%
Sprendimo taškas (angl. *Decision*)	100%	100%	-	-
Procesas (angl. *Process*)	100%	96.4%	-	-
Baigimo taškas (angl. *Terminator*)	100%	100%	-	-
Tekstas	98.3%	99.5%	98.5%	-

Bendras tikslumas	98.4%	99%	99.1%	97%

1.8 lentelė. *Arrow R-CNN* modelio simbolių atpažinimo tikslumas *FA* duomenų rinkiniui.

Klasė	**Tikslumas (FA)**
Rodyklė (angl. *Arrow*)	99%
Galutinė būsena (angl. *Final state*)	100%
Būsena (angl. *State*)	100%
Tekstas	99.6%
Bendras tikslumas	99.5%

1.9 lentelė. Arrow R-CNN modelio diagramų atpažinimo tikslumas skirtingiems duomenų rinkiniams

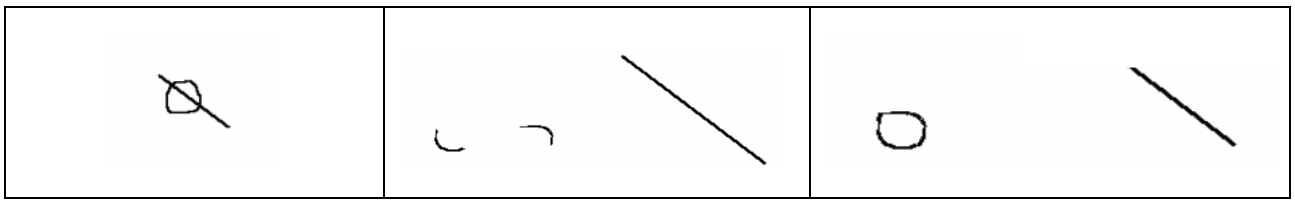
	FC_A	**FC_B**	**FA**	**DIDI**
Tikslumas	68.4%	78.6%	83.3%	83.9%

Iki šio sprendimo pasiūlymo, geriausias diagramų atpažinimo tikslumas iš eskizų *FC_B* srautų diagramų duomenų rinkiniui buvo 37.7%. Naujesnis *DrawNet* sprendimas pasiekia apytiksliai ~2% geresnį tikslumą *FC_A, FC_B, FA* duomenų rinkiniams negu *Arrow R-CNN.*

1.5.3. Atraminių vektorių metodas SVM

Diagramų elementų atpažinimui iš eskizų naudojamos ne tik CNN giliojo neuroninio tinklo architektūros, tačiau ir kiti mašininio mokymosi metodai. Vienas iš tokių metodų šiai problematikai spręsti yra SVM (1.3.2.1) atraminių vektorių metodas [63]. Pasiūlytame sprendime diagramos atpažinimas susideda iš 4 pagrindinių žingsnių: eskizo segmentavimo, ypatybių vektorių išgavimo, diagramos simbolių klasifikavimo ir diagramos skaitmenizavimo.

Segmentavimą galima suskaidyti į 3 pagrindinius žingsnius: priekinio plano pikselių atskyrimą nuo antrinio plano pikselių atskyrimo, teksto atskyrimą iš eskizo, linijos primityvų išgavimą iš priekinio plano pikselių. Priekinio plano pikselių atskyrimui nuo antrinio plano pikselių yra naudojamas adaptyvus slenksčio taikymas (1.22 pav.), o teksto atskyrimui naudojamas dydžio filtras, nes buvo pastebėta, kad teksto komponentės reliatyviai turi mažesnį dydį negu kiti grafiniai elementai eskize. Sudėtingiausias ir svarbiausias segmentavimo žingsnis yra linijos primityvų atskyrimas, nes tai nulemia kokybę ir tikslumą klasifikavimo žingsnyje. Linijos atskyrimui yra naudojamas linijos primityvų išskyrimo interpretuojant linijos tęsinį (angl. *Line Primitive Extraction by Interpretation of Line Continuation)* metodas. Šiuo metodu matematiškai modeliuojamos dvi formos savybės: lygumas (angl. *smoothness)* ir vientisumas (angl. *continuity*), kurias naudoja žmonės, norėdami iš diagramos išgauti formą. Šio metodo rezultatai atskiria linijos primityvus, tačiau gali nekorektiškai atskirti elementą į skirtingas linijas, nes nėra atsižvelgiama į diagramos elementų kontekstą. Analizuojamas sprendimas papildo šį metodą, pridedant segmentavimo klaidų korekcijos modulį, kuris atsižvelgia į diagramos kontekstą ir sujungia susijusius atskirtus linijos tęsinius bei sugrupuoja gautus rezultatus į linijas ir kitus diagramos simbolius. Objektai yra sugrupuojami pagal tai ar savyje turi skylių. Linijos ir diagramos simboliai yra sujungiami tarpusavyje pagal tai kaip šių elementų galai yra arti vienas kito.



1.33 pav. Kairėje originali nuotrauka; per vidurį gautas rezultatas pritaikius linijos primityvų atskyrimą interpretuojant linijos tęsinį metodas; dešinėje gautas rezultatas su papildomu korekcijos moduliu [63].

Ypatybių vektoriui sudaryti yra naudojamas objekto (angl. *convex hull*) plotas (*Ach*); perimetras (*Pch*); kvadrato plotas, ribojantis objekto perimetrą (*Aer*); maksimalaus kvadrato, telpančio į objekto vidų, plotas (*Alq*); maksimalaus trikampio, telpančio į objekto vidų, plotas (*Alt*); trikampio perimetras (*Plt*); kvadrato, gaubiančio objekto perimetrą aukštis (*Her*) ir plotis (*Wer*). Iš šių bazinių ypatybių vektorių papildomai sudaromi išvestiniai duomenys padedantys klasifikuoti diagramos simbolius:

- Pch^2/Ach - plonumo santykis (angl. *thiness ratio*). Apskritimo plonumo santykis yra minimalus, nes tai plokštumos figūra su mažiausiu perimetru, apimančiu tam tikrą plotą, todėl šis santykis padeda išskirti apskritimus.
- Ach/Aer ir Alq/Aer - stačiakampiams šis santykis turėtų būti artimas vienetui, todėl tai padeda atskirti stačiakampius nuo kitų figūrų.
- Alq/Ach - santykis, padedantis atskirti elipses nuo kitų figūrų. Elipsėms šis santykis yra apie 0.7, o kitoms figūroms didesnis.
- Alt/Alq – santykis, padedantis atskirti deimanto formos figūras. Dažniausiai šio tipo figūroms santykis yra tarp 0.5 ir 0.6.
- Alt/Ach ir Plt/Pch - trikampiams šie santykiai turėtų būti artimi vienetui, todėl tai padeda atskirti trikampius nuo kitų figūrų.
- Her/Wer – santykis, padedantis atskirti linijas nuo kitų figūrų.

Sudarytiems ypatybių vektoriams yra taikomas šešių dvejetainių klasių SVM klasifikatorius su technika „vienas prieš kitą“ (angl. *one versus the rest*). Kiekviena klasė yra susieta su viena iš šešių figūrų: apskritimu, trikampiu, stačiakampiu, deimantu, elipse, linija. Kiekvienas klasifikatorius grąžina gautą patikimumo lygį (angl. *confidence level*) susietai klasei, pavyzdžiui, stačiakampio klasifikatorius pateikia atsakymą, kokia tikimybė, kad pateikta figūra yra stačiakampis. Visuose dvejetainiuose klasifikatoriuose kaip branduolys naudojama radialinė bazinė funkcija, kurios parametras gama lygus 2.

1.10 lentelė. Diagramos simbolių atpažinimo tikslumas pritaikius SVM klasifikatorių

	Apskritimas	**Trikampis**	**Stačiakampis**	**Deimantas/Sprendimo taškas**	**Elipsė**	**Linija**	**Bendras**
Tikslumas	100%	95%	65%	95%	85%	100%	90%

Apmokymo imtis buvo sudaryta iš 720 diagramos simbolių, po 120 kiekvienam simboliui, o testavimo imtis atitinkamai iš 120 ir 20. Gautas bendras pavienių simbolių atpažinimo tikslumas siekia 90%, tačiau analizuojamame sprendime nėra pateiktas bendras diagramos simbolių sujungimo tikslumas.

1.5.4. Esamų problemos sprendimų analizė

Esamų problemos sprendimų palyginimo kriterijai:

– Diagramų atpažinimo tikslumas – bendras diagramų atpažinimo tikslumas, kai visi diagramos simboliai yra atpažinti ir sujungti teisingai;
– Diagramų konvertavimas į XMI formatą – ar sprendimo sąsaja suteikia būdą konvertuoti skaitmenizuotą diagramą į XMI formato failą tolimesniam diagramos tobulinimui;
– Mašininio mokymosi metodas – koks mašininio mokymosi metodas yra naudojamas diagramos simbolių atpažinimui;
– Diagramos skaitmenizavimo sąsajos prieinamumas – ar prieinama diagramos skaitmenizavimo sąsaja naudotojui;
– Teksto skaitmenizavimas – ar atpažintas ranka rašytas tekstas yra paverčiamas į skaitmenizuotą formatą;
– Pavienių diagramos simbolių tikslumas – atskirų diagramos elementų atpažinimo tikslumas.

1.11 lentelė. DrawNet, *Arrow R-CNN* ir sprendimo naudojant SVM metodą, palyginimas

	DrawNet	Arrow R-CNN	SVM
Diagramų atpažinimo tikslumas	Aukštas (~70%)	Aukštas (~68%)	-
Diagramų konvertavimas į XMI formatą	-	-	-
Mašininio mokymosi metodas	CNN	CNN	SVM
Atpažįstamos diagramos	Srauto diagrama, būsenų diagrama	Srauto diagrama, būsenų diagrama	Dalinai srauto ir būsenų diagrama
Diagramos skaitmenizavimo sąsajos prieinamumas	-	-	-
Atpažįstama notacija	UML	BPMN	-
Teksto skaitmenizavimas	-	-	-
Pavienių diagramos simbolių tikslumas			
Rodyklė (angl. *Arrow*)	95.7% - 98.6%	95.2% - 99%	-
Jungtis (angl. *Connection*)	~100%	99.6%	100%
Duomenys (angl. *Data*)	~100%	~100%	-
Baigimo taškas (angl. *Terminator*)	~100%	100%	-
Sprendimo taškas (angl. *Decision*)	100%	100%	95%
Procesas (angl. *Process*)	100%	96.4% - 99%	-
Būsena (angl. *State*)	100%	100%	-
Galutinė būsena (angl. *Final state*)	100%	100%	-
Tekstas	98.3%	98.3% - 99.6%	-

Trikampis	-	-	95%
Apskritimas	-	-	100%
Elipsė	-	-	85%
Bendras pavienių simbolių tikslumas	98.4% - 99.5%	98.4% - 99.5%	90%

Analizuoti *DrawNet* ir *Arrow R-CNN* sprendimai yra tarpusavyje panašūs. Abu siekia gana aukštą diagramų atpažinimo tikslumą ~70% bei naudoja CNN neuroninio tinklo architektūrą. Galima pastebėti, kad rinkoje egzistuojantys ir prieinami sprendimai, kaip *R-CNN, Faster R-CNN* nesuteikia iš karto gerų rezultatų, todėl analizuotų sąsajų autoriai turėjo patobulinti neuroninio tinklo architektūras, jas praplėsdami papildomais moduliais. *DrawNet* aukštą tikslumą pasiekti padėjo papildomų trijų, kairiojo viršutinio kampo, dešiniojo apatinio kampo ir rodyklės esminių taškų nustatymo modulių pridėjimas. *Arrow R-CNN* sprendimas praplėtė *Faster R-CNN* modelio veikimą pridedant, kaip ir *DrawNet* atveju, papildomą rodyklės esminių taškų nustatymo modulį. Abiem atvejais tikslumą pagerinti padėjo duomenų praplėtimo strategijos. *DrawNet* ir *Arrow R-CNN* sąsajos gali atpažinti srauto ir baigtinių būsenų diagramas, tačiau nepateikia būdo, kaip skaitmenizuotas diagramas paversti į XMI formatą diagramos pasidalinimui ir tolimesniam redagavimui. Taip pat, šių sąsajų autoriai apsiriboja diagramos simbolių skaitmenizavimu ir nesprendžia ranka rašyto teksto atpažinimo ir skaitmenizavimo užduočių. Galima pastebėti, kad pavienių diagramos simbolių atpažinimas yra bene lengviausia dalis diagramų skaitmenizavimo problematikoje, nes pateikti sprendimai siekia nuo 90% (SVM) iki 99.5% (CNN) tikslumą. Sunkiausia yra atpažintus simbolius sujungti tarpusavyje, sudarant galutinę diagramos struktūrą. Tai pasiekti padeda rodyklių orientacijos bei pradžios ir pabaigos taškų nustatymas. Taip pat, atliekant literatūros analizę buvo pastebėta, kad Lietuvoje diagramų atpažinimo ir skaitmenizavimo problematikoje jokių tyrimų nebuvo atlikta.

1.6. Analizės išvados

Analizės metu buvo apžvelgta UML kalba ir jos taikymas sistemoms projektuoti, išanalizuoti CNN ir SVM mašininio mokymosi metodai vaizdų atpažinime, apžvelgtos teksto atpažinimo galimybės bei išanalizuotos egzistuojančios diagramų eskizų skatmenizavimo sąsajos.

1. Nors šiuo metu rinkoje egzistuoja daug įrankių leidžiančių projektuoti UML panaudojimo atvejų diagramas, praktikoje dažnai pirmieji diagramų eskizai yra nubraižomi ranka ant popieriaus arba išmaniosios lentos (angl. *white board*) diskusijų metu tarp sistemos kūrėjų ir kitų suinteresuotų asmenų. Norint efektyviai pasidalinti eskizu su kitais suinteresuotais asmenimis bei toliau plėtoti pradinį diagramos eskizą, pastarąjį reikia rankiniu būdu perkelti į UML standartą palaikantį įrankį. Rankinis diagramos perkėlimas yra neefektyvus, tačiau nėra įrankio, kuris leistų automatiškai skaitmenizuoti UML panaudojimo atvejų diagramą iš eskizo.
2. Ranka braižytų diagramų atpažinimo metodikos literatūroje skirstomas į dvi pagrindines kategorijas: realaus laiko braižomų diagramų (angl. *online-targeted*) ir eskizų (angl. *offline-targeted)* atpažinimą ir skaitmenizavimą. Realaus laiko braižomų diagramų atpažinimo metodai negali būti sėkmingai pritaikyti diagramų eskizų skaitmenizavimui dėl tokių metodų priklausomybės nuo istorinių diagramos braižymo duomenų, kurie eskizuose neegzistuoja. Diagramų eskizų skaitmenizavimui dažniausiai naudojami CNN ir SVM mašininio mokymosi metodai. Geriausias pasiektas diagramos atpažinimo tikslumas iki šiol yra apie ~70% (CNN).

Verta paminėti, kad analizuoti sprendimai negali atlikti pilno diagramos skaitmenizavimo ir eksportavimo, nes nėra skaitmenizuojamas tekstas. Atlikta teksto atpažinimo (OCR) galimybių analizė padėjo suprasti, kad ranka rašyto teksto atpažinimas ir skaitmenizavimas yra sudėtingas uždavinys, nes reikalinga sukurti atskirą OCR sprendimą lietuviškiems simboliams ir tekstui atpažinti.

3. Egzistuojančios diagramų skaitmenizavimo sąsajos siekia aukštus diagramos atpažinimo rezultatus, tačiau sprendimai nėra išplėtoti ir prieinami galutiniam naudotojui. Sprendimai yra pritaikyti standartizuotų srautų (angl. *flowchart*) ir būsenų (angl. *finite automata*) diagramų skaitmenizavimui, bet nepalaiko UML panaudojimo atvejų (angl. *Use Case*) diagramų atpažinimo galimybių, diagramos eksportavimo į XMI formatą. Analizuoti sprendimai neturi įrankio leidžiančio atlikti diagramos eskizo skaitmenizavimo procesą galutiniam naudotojui.

2. Sprendimo projektas ir formalizuotas aprašas

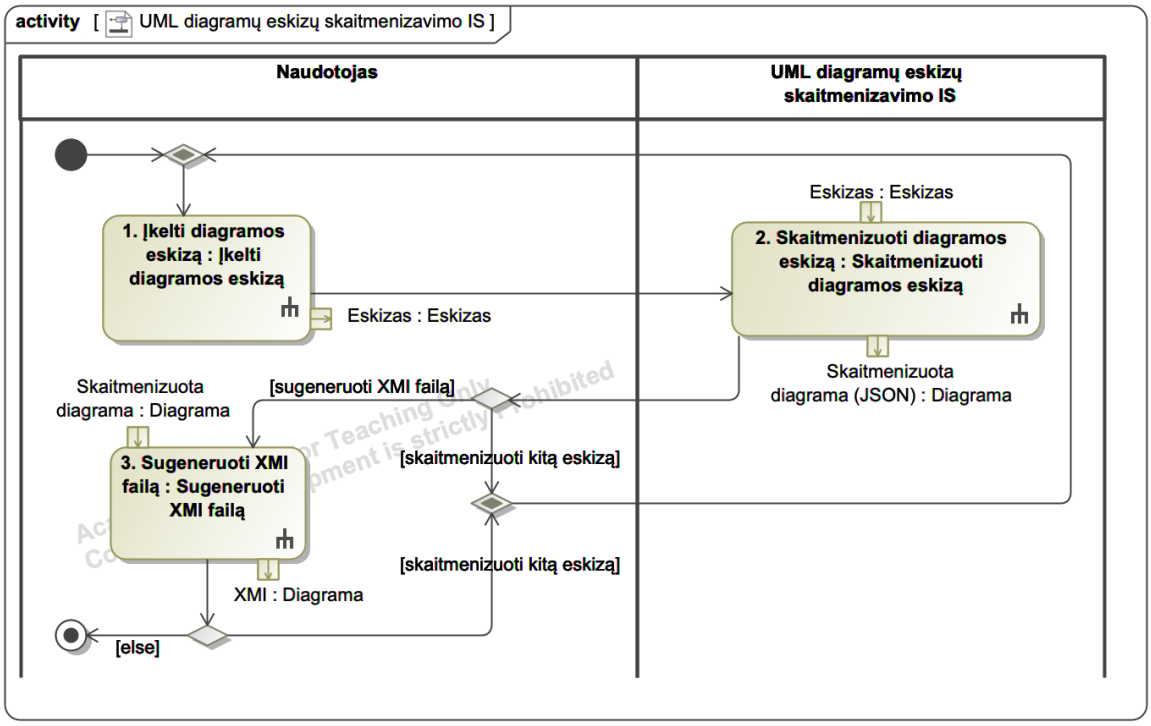
Šiame skyriuje pateikiamas sprendimo projektas ir formalizuotas aprašas. 2.1 poskyryje aprašyti pagrindiniai sistemos funkciniai reikalavimai, pavaizduoti panaudojimo atvejų ir veiklos diagramose. 2.2 poskyryje aprašytas informacinės sistemos esybių klasių modelis su pagrindinėmis dalykinės srities sąvokomis. 2.3 poskyryje pateiktas sprendimo formalizuotas aprašas, kuriame detalizuojami UML diagramų eskizų skaitmenizavimo IS pagrindiniai komponentai ir veikimo principai.

2.1. Funkciniai reikalavimai

Sistemos pagrindinės funkcijos yra pavaizduotos UML panaudojimo atvejų diagramoje ir detalizuotos UML veiklos diagramose, kuriuose parodoma, kokius veiksmus atlieka sistemos aktoriai ir kaip reaguoja sistema. Pagrindinis diagramų eskizų skaitmenizavimo procesas ir panaudojimo atvejų veiklos diagramos yra vaizduojamos ir aprašomos (2.2 pav – 2.6 pav.) Pagrindiniai sistemos aktoriai yra:

- *Naudotojas*, turintis prieigą prie viso sistemos funkcionalumo. Kiekvienas sistemos naudotojas gali įkelti diagramos eskizą, peržiūrėti skaitmenizuotų diagramų istoriją ir skaitmenizuoti diagramos eskizą. Skaitmenizavus diagramą, naudotojas gali parsisiųsti XMI failą;
- *Google Cloud Vision OCR*, išorinis sistemos aktorius, kuris teikia OCR paslaugą diagramos skaitmenizavimo metu.

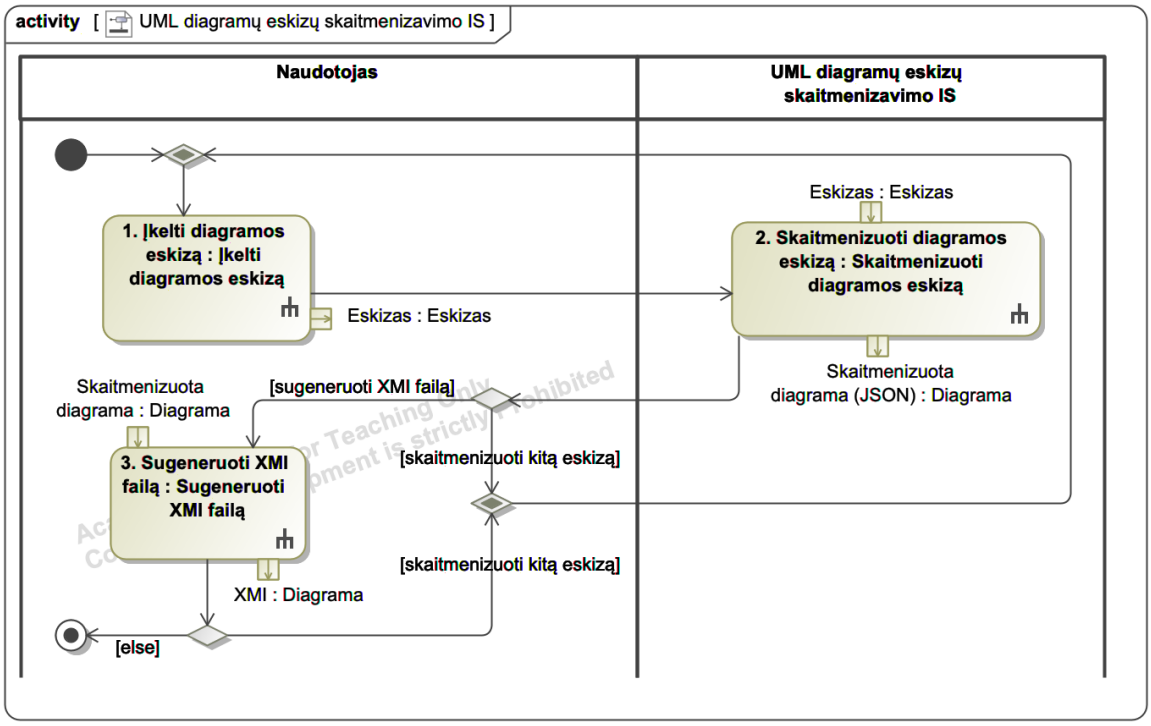
2.1 pav. UML diagramų eskizų skaitmenizavimo IS panaudojimo atvejų diagrama.



2.2 pav. Pagrindinis diagramos eskizo skaitmenizavimo proceso PA

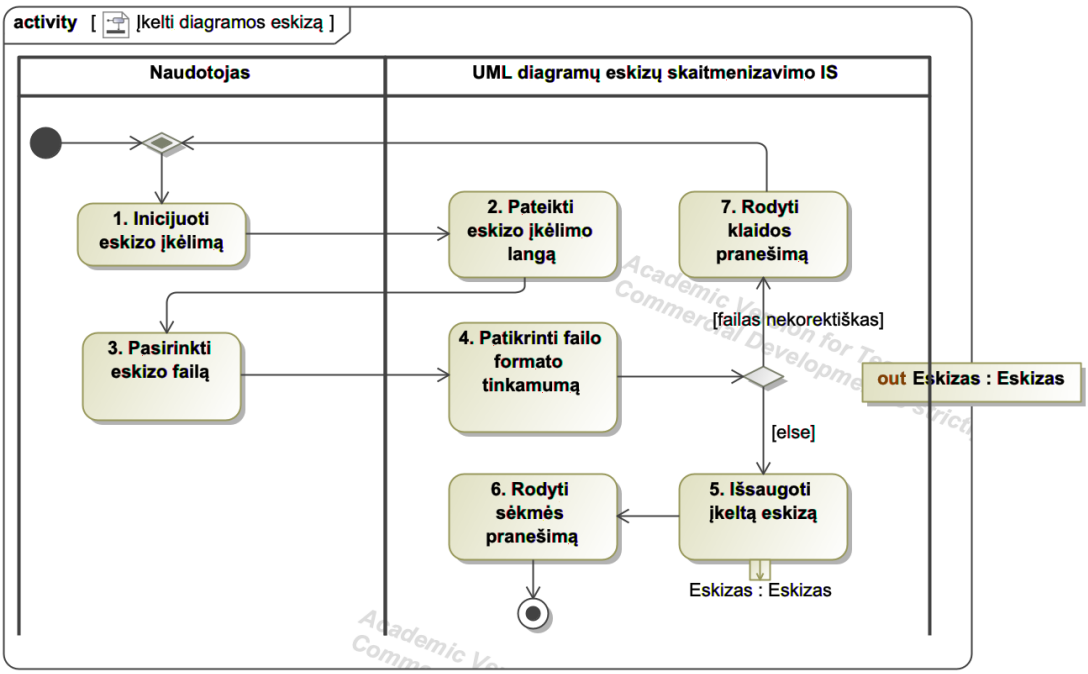
2.2 pav. yra pavaizduotas pagrindinis skaitmenizavimo procesas iš naudotojo perspektyvos. Naudotojas yra pagrindiniame diagramos eskizo skaitmenizavimo lange ir įkelia diagramos eskizą. Iniciavus diagramos skaitmenizavimą sistema skaitmenizuoja diagramą ir pateikia rezultatą naudotojui. Toliau naudotojas gali:

- baigti darbą;
- skaitmenizuoti kitą diagramą;
- skaitmenizuotos diagramos rezultatą parsiųsti XMI failo formatu, kurį būtų galima įsikelti į XMI failo formatą palaikantį CASE įrankį ir pratęsti modeliavimą.



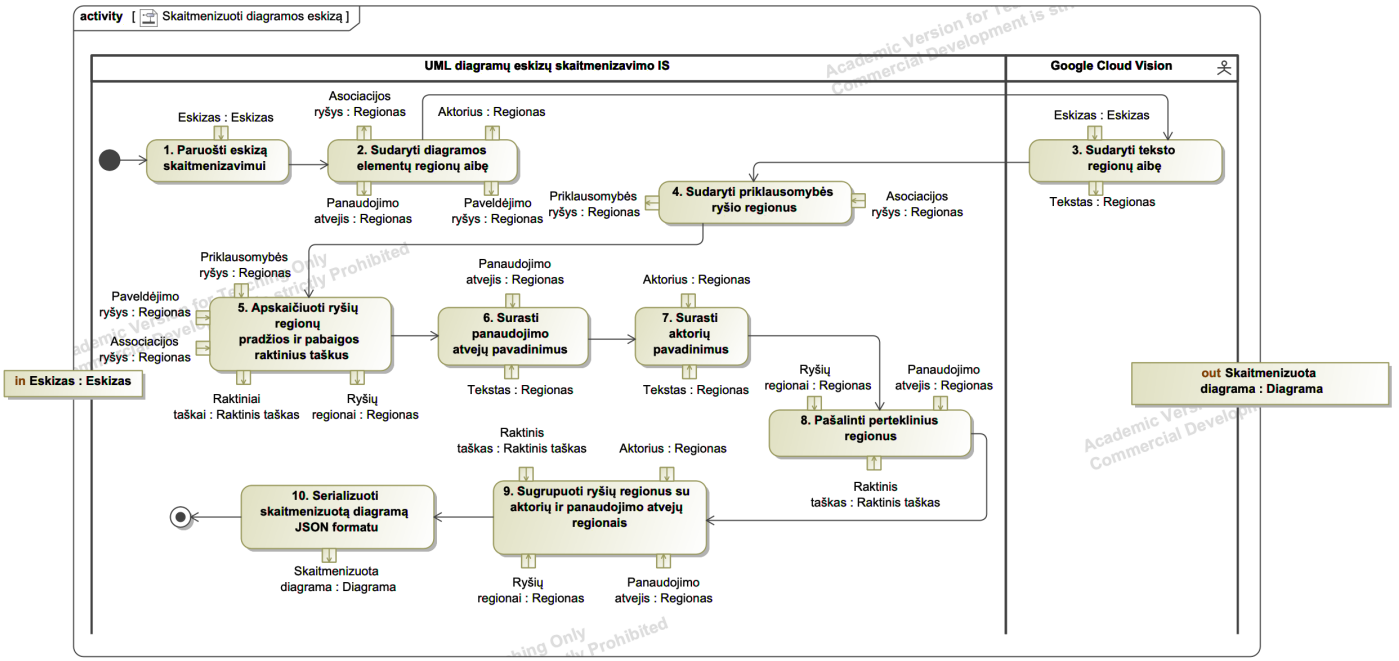
2.2 pav. Pagrindinis diagramos eskizo skaitmenizavimo proceso PA

2.3 pav. yra pavaizduota „Įkelti diagramos eskizą“ panaudojimo atvejų veiklos diagrama. Naudotojas inicijuoja eskizo įkėlimo veiksmą. Sistemai pateikus failo įkėlimo langą, naudotojas pasirenka norimą įkelti eskizą. Sistema patikrina failo korektiškumą. Jeigu failas yra nekorektiškas, naudotojui yra parodomas klaidos pranešimas ir failą įkelti galima bandyti dar kartą. Kitu atveju, diagramos eskizo failas yra išsaugomas sistemoje ir parodomas sėkmės pranešimas. Toliau naudotojas įkeltą diagramos eskizą gali skaitmenizuoti.



2.3 pav. „Įkelti diagramos eskizą“ panaudojimo atvejų veiklos diagrama

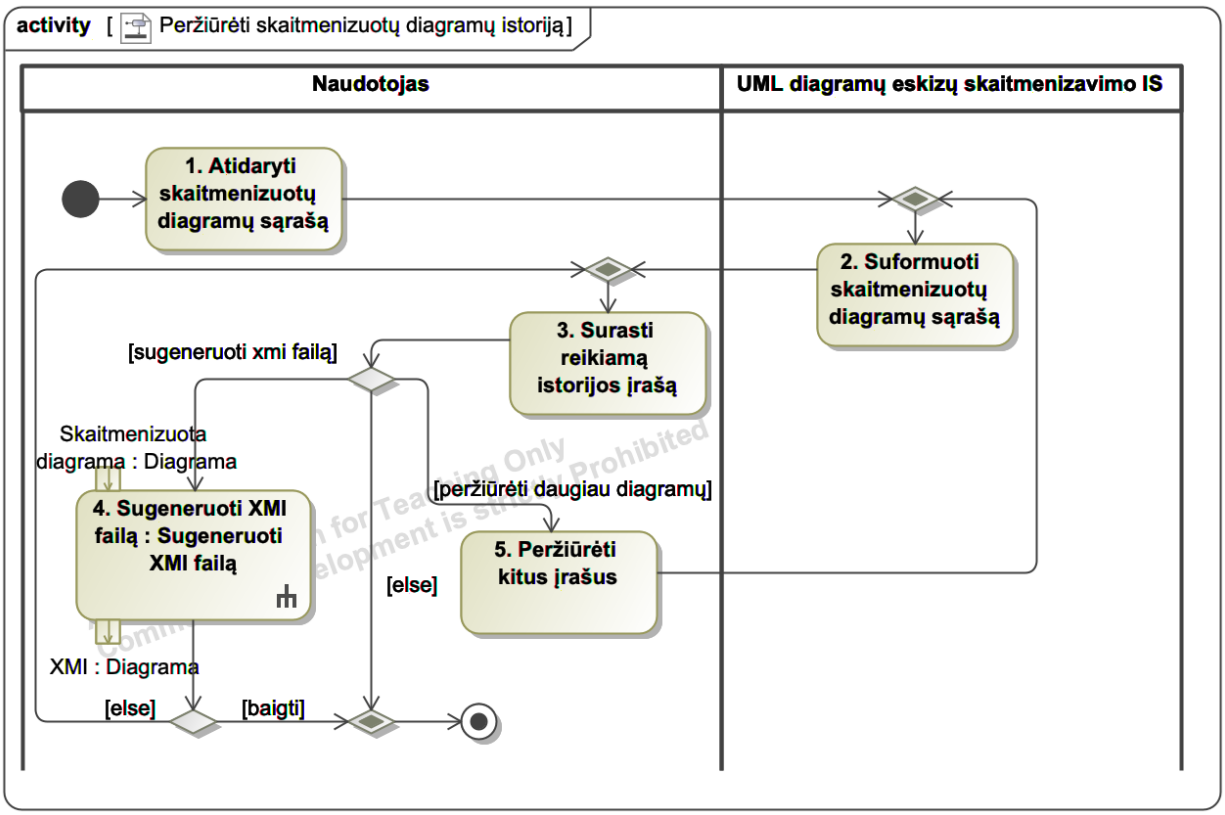
2.4 pav. yra pavaizduota „Skaitmenizuoti diagramos eskizą“ panaudojimo atvejų veiklos diagrama. Šis procesas yra vykdomas, kai naudotojas inicijuoja įkelto diagramos eskizo skaitmenizavimą. Sistema paruošia eskizą skaitmenizavimui atlikdama išankstinio apdorojimo veiksmus (keičiamas eskizo dydis reikalingas klasifikatoriui, panaikinami nereikalingi paveiksliukų sluoksniai). Iš eskizo sudaromas požymių vektorius, kuris pateikiamas CNN simbolių atpažinimo modeliui. CNN modelis atlieka išvados etapą (angl. *inference*), kurio metu sudaro asociacijos ryšių, paveldėjimo ryšių, aktorių ir panaudojimo atvejų elementų regionų aibę (2.3.1). Toliau diagramos eskizas yra pateikiamas *Google Cloud Vision OCR* paslaugai, kurios rezultatas yra teksto regionų aibė (2.3.3). Po OCR žingsnio vykdomas priklausomybės ryšio regionų nustatymas iš asociacijos ryšio regionų. Visiems ryšių regionams nustatomi pradžios ir pabaigos raktiniai taškai (2.3.2). Nustačius raktinius taškus, surandami panaudojimo atvejų ir aktorių pavadinimai. Atliekamas papildomas panaudojimo atvejų ir ryšių regionų apdorojimas, kurio metu pašalinami pasikartojantys regionai. Toliau ryšių regionai yra sugrupuojami su aktorių ir panaudojimo atvejų regionais pagal ryšių regionų pradžios ir pabaigos raktinius taškus. Galiausiai iš turimos informacijos yra sudaroma skaitmenizuotos diagramos struktūra JSON formatu (2.3.4).



2.4 pav. „Skaitmenizuoti diagramos eskizą“ panaudojimo atvejų veiklos diagrama

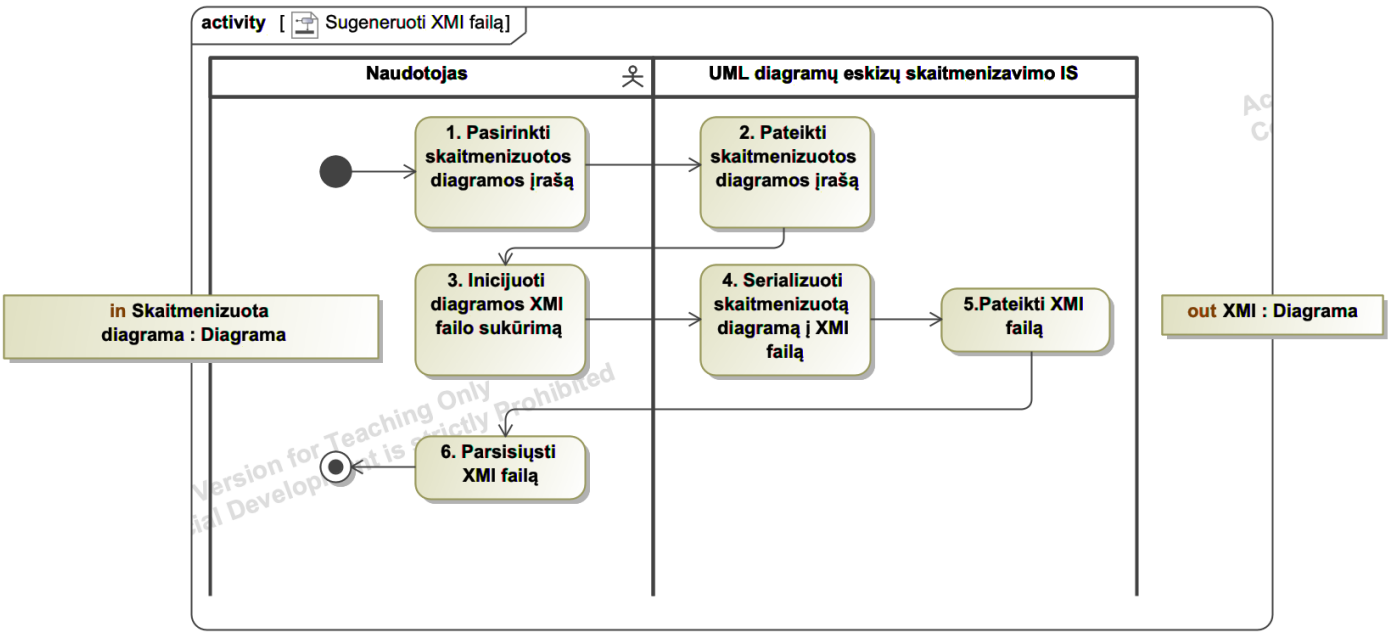
2.5 pav. yra pavaizduota „Peržiūrėti skaitmenizuotų diagramų istoriją“ panaudojimo atvejų veiklos diagrama. Naudotojas pagrindiniame sistemos lange atidaro skaitmenizuotų diagramų istorijos sąrašą. Sistema suformuoja skaitmenizuotų diagramų istorijos sąrašą ir pateikia naudotojui. Toliau gali būti vykdomi tokie veiksmai:

– jeigu naudotojas nerado norimos diagramos, gali pasirinkti peržiūrėti kitus skaitmenizuotų diagramų istorijos įrašus;
– sugeneruoti pasirinktos skaitmenizuotos diagramos XMI failą (2.6 pav.).



2.5 pav. „Peržiūrėti skaitmenizuotų diagramų istoriją“ panaudojimo atvejų veiklos diagrama

2.6 pav. yra pavaizduota „Sugeneruoti XMI failą“ panaudojimo atvejų veiklos diagrama. Naudotojas pasirenka norimą skaitmenizuotos diagramos įrašą, kurį pateikia sistema. Toliau inicijuojamas diagramos XMI failo parsiuntimas. Sistema serializuoja skaitmenizuotą diagramą ir sukuria XMI failą (2.3.5). Sistema pateikia naudotojui sukurtą XMI failą ir inicijuoja failo parsiuntimą.

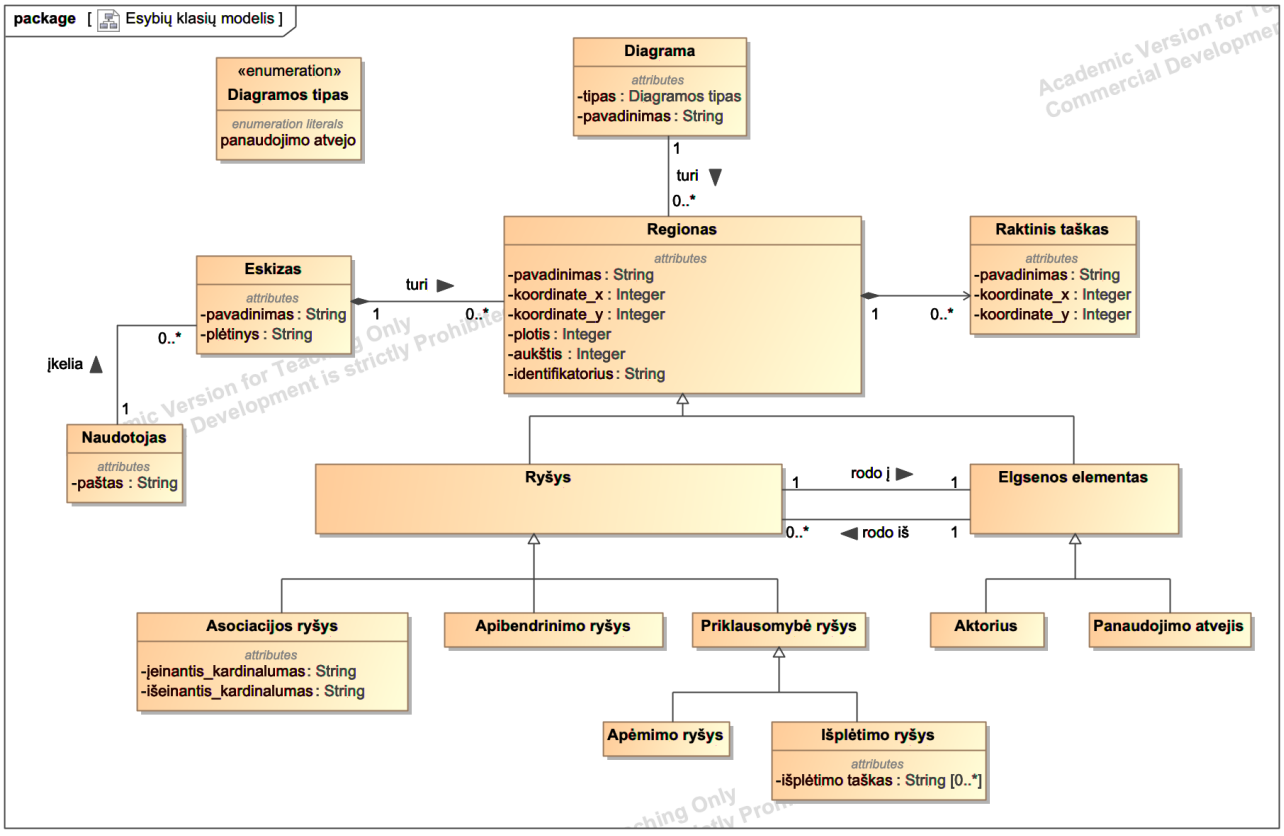


2.6 pav. „Sugeneruoti XMI failą“ panaudojimo atvejų veiklos diagrama

Šiame poskyryje buvo apžvelgtos UML diagramų eskizų skaitmenizavimo IS pagrindinės funkcijos pavaizduotos UML ir panaudojimo atvejų diagramose (2.2 pav. – 2.6 pav.). Kitame poskyryje (2.2) aprašomas dalykinės srities modelis apžvelgiant pagrindines dalykinės srities sąvokas.

2.2. Dalykinės srities modelis

2.7 pav. yra pavaizduotas UML diagramų eskizų informacinės sistemos esybių klasių modelis. Kiekvienas sistemos naudotojas gali į sistemą įkelti diagramų eskizus, kurie gali būti skaitmenizuoti, kaip pateikta 2.4 pav. Eskizas susideda iš pavadinimo ir plėtinio. Po eskizo skaitmenizavimo, kiekvienas diagramos simbolis turi savo regioną eskize. Regionas susideda iš regiono pavadinimo, koordinačių ir dydžio. Tam tikri regionai, pavyzdžiui, asociacijų ryšio regionai, turi papildomus raktinius taškus, kad būtų galima nustatyti jų kryptį regione. Raktiniai taškai turi pavadinimą ir koordinates. Bendra regionų visuma sudaro diagramą. Kiekvienas regionas atitinka panaudojimo atvejų diagramos simbolį: ryšio elementą arba elgsenos elementą. Ryšio elementai yra išskirstyti į asociacijos, apibendrinimo ir priklausomybės ryšius. Asociacijos ryšiai papildomai turi susietus kardinalumus ryšio galuose. Priklausomybės ryšiai savo ruožtu yra išskaidyti į apėmimo (angl. *include*) ir išplėtimo (angl. *extend*) ryšius. Išplėtimo ryšys turi priskirtus išplėtimo taškus – sąlygas. Elgsenos elementai yra specifikuojami į aktoriaus ir panaudojimo atvejų elementus. Iš vieno elgsenos elemento gali būti daug išeinančių ryšio elementų. Vienas ryšio elementas, gali rodyti tik į vieną elgsenos elementą.



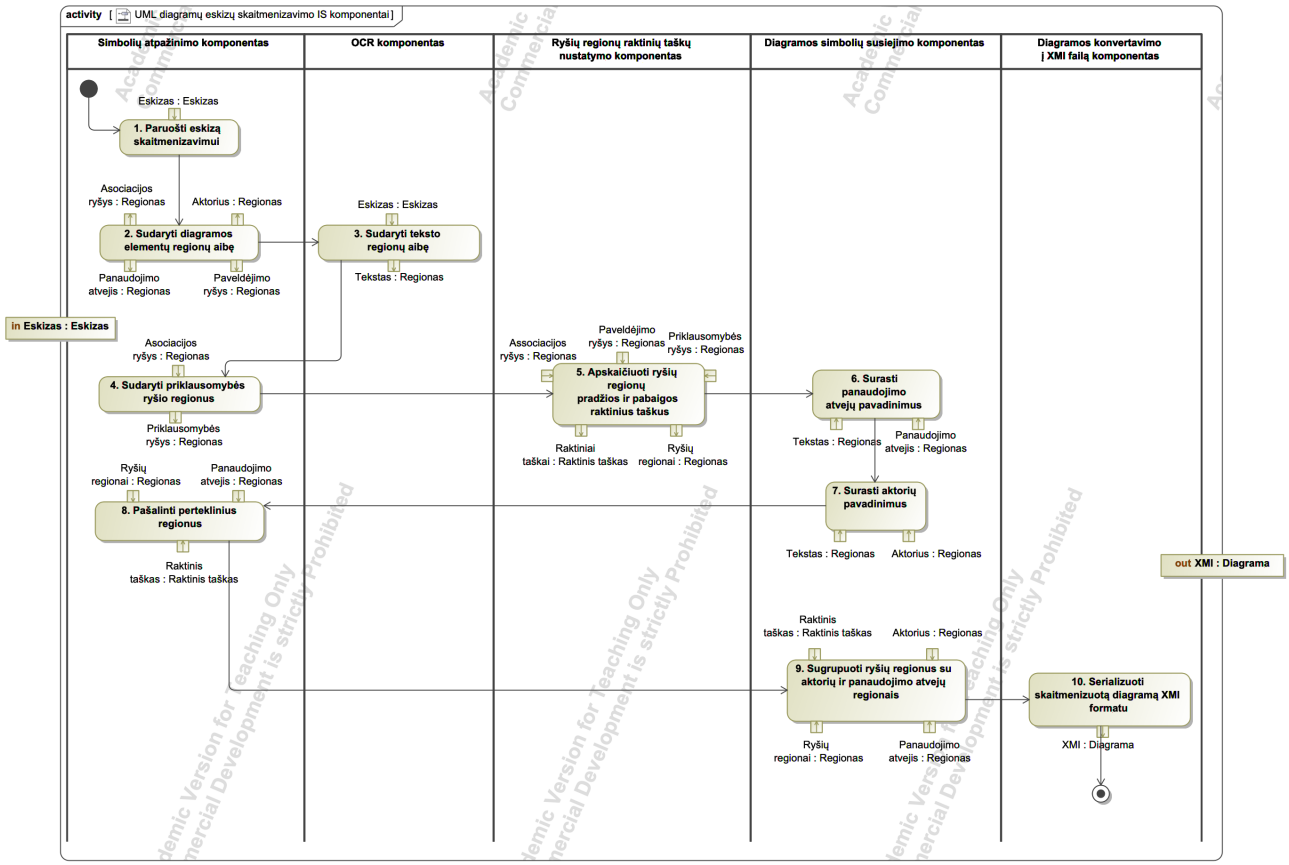
2.7 pav. UML diagramų eskizų skaitmenizavimo IS esybių klasių modelis

Šiame poskyryje buvo apžvelgtas UML diagramų eskizų skaitmenizavimo IS dalykinės srities modelis pavaizduotas klasių diagrama. Kitame poskyryje (2.3) pateiktas sprendimo formalizuotas aprašas, kuriame detalizuojami pagrindiniai sistemos komponentai ir jų veikimo principai.

2.3. Formalizuotas sprendimo aprašas

2.8 pav. yra pavaizduoti UML diagramų eskizų skaitmenizavimo IS komponentai ir eskizo skaitmenizavimo seka. Pagrindiniai sistemos komponentai yra:

- Diagramos simbolių atpažinimo komponentas – eskize suranda diagramos elementų regionų aibę;
- Ryšių regionų raktinių taškų nustatymo komponentas – ryšių regionams nustato pradžios ir pabaigos raktinių taškų koordinates;
- OCR komponentas – eskize suranda teksto regionus;
- Diagramos simbolių susiejimo komponentas – susieja atskirus diagramos elementus į bendrą struktūrą;
- Diagramos konvertavimo į XMI failą komponentas – konvertuoja skaitmenizuotą diagramą į XMI failą.



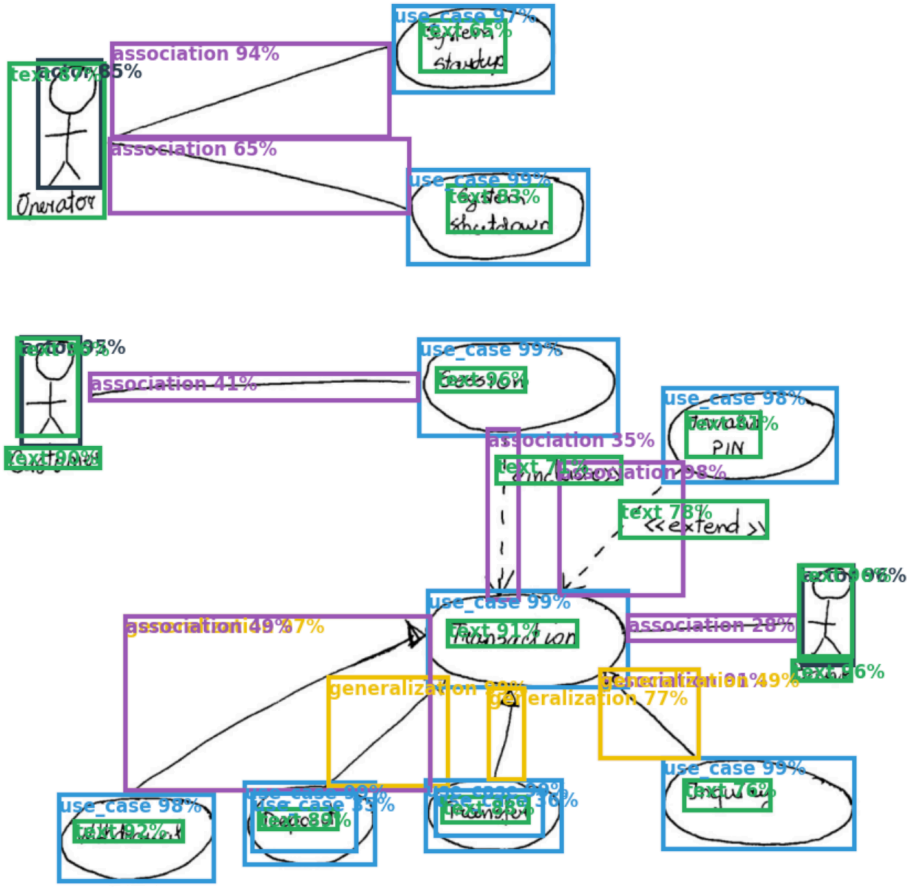
2.8 pav. Diagramos eskizų skaitmenizavimo sistemos pagrindiniai komponentai

Diagramos eskizas yra apdorojamas diagramos simbolių atpažinimo komponente, kuris pateikia regionų aibę. Teksto ir ryšių regionai yra atitinkamai apdorojami ryšių regionų raktinių taškų nustatymo komponente ir OCR komponente. Toliau visi regionai yra pateikiami diagramos simbolių susiejimo komponentui. Šio komponento rezultatas yra konvertuojamas į XMI failą.

2.3.1. Diagramos simbolių atpažinimo komponentas

Diagramos simbolių atpažinimo komponente siekiama aptikti visus diagramos simbolių regionus, kurie patenką į keturias klases: *association* – asociacijos ir priklausomybės ryšiai, *generalization* – paveldėjimo ryšys, *actor* – sistemos aktorius, *use case* – panaudojimo atvejas. Šio komponento įvestis

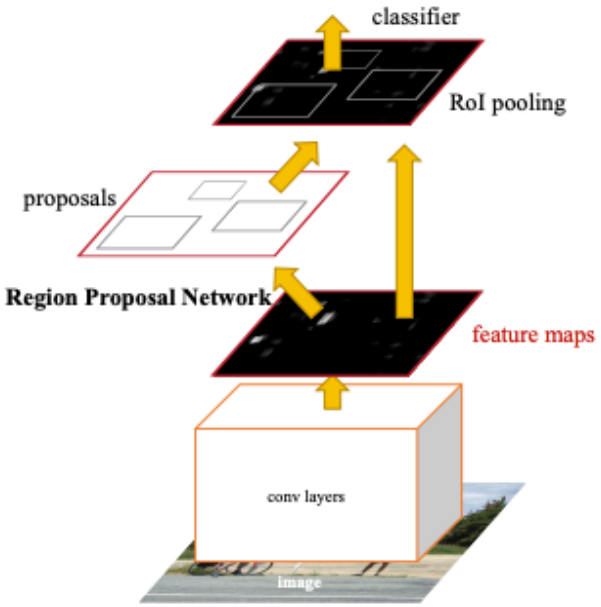
yra diagramos eskizas, kurio dydis pakeičiamas taip, kad ilgesnioji kraštinė būtų lygi 1 333 taškams, t. y. standartinis formatas, kurio reikalauja *Faster R-CNN ResNet* neuroninis tinklas, kuris naudojamas šiame etape. Šio etapo rezultatai yra aptikti ir atrinkti objektai - regionai *R*, kur kiekvienas regionas yra nusakomas trimis reikšmėmis (*b*, *c, s*). Reikšmė *b yra* objektą ribojantis langas (angl. *bounding box*), *c* yra objektą nusakanti klasė (angl. *label*), o *s* reikšmė yra skaliaras, nusakantis *c* reikšmės spėjimo patikimumą (angl. *prediction score*). 2.9 pav. yra pavaizduotas diagramos simbolių atpažinimo komponento išvesties rezultatas. Svarbu paminėti, kad pavyzdyje matomi teksto regionai yra sudaromi po OCR komponento žingsnio.



2.9 pav. Diagramos simbolių atpažinimo komponento rezultatas

Diagramos simbolių aptikimo komponentui realizuoti naudojamas *Faster R-CNN* tinklo *TensorFlow Object Detection 2* bibliotekos realizacija su pritaikyta apmokymo konfigūraciją pagal UML 2.5 panaudojimo atvejų diagramos simbolius ir kontekstą. *Faster R-CNN* yra populiarus gilusis neuroninis tinklas naudojamas objektams aptikti nuotraukoje, ir kuris grąžina regionius *R*, kur kiekvienas iš jų turi ankščiau apibūdintas reikšmes (*b, c, s*). *Faster R-CNN* tinklas dirba su RGB paveikslėliais, kurie nusakomi trimačiu masyvu: pirmosios dvi dimensijos yra paveiksliuko dydis pikseliais. Paskutinioji dimensija yra vektorius sudarytas iš trijų reikšmių. Kiekviena reikšmė nusako pagrindinės spalvos: raudonos, žalios ir mėlynos spalvos intensyvumą. *Faster R-CNN* aptinka objektus pagal paveikslėlio ypatybių vektorių (angl. *feature maps*), kuris yra sudarytas pilnai jungto CNN bazinio tinklo (angl. *backbone*). Paveikslėlio ypatybių vektorius po šio žingsnio yra papildomas ne tik paveikslėlio dydžio ir RGB spalvų intensyvumo reikšmėmis, bet ir įvarių dėsningumų, išreikštų

pikseliuose. Pavyzdžiui, tam tikruose paveikslėlio regionuose pastebimi dėsningumai, kaip braižoma įstrižainė linija, rodyklės galvutė ar aktorius.



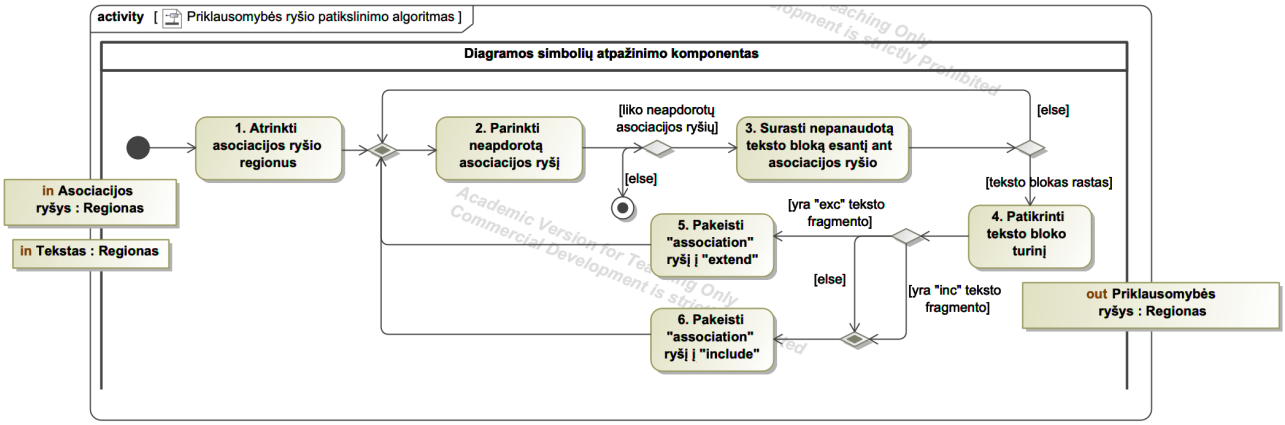
2.10 pav. *Faster R-CNN* dviejų etapų metodas sudarytas iš CNN bazinio sluoksnio pateikiančio ypatybių vektorių, RPN tinklo sudarančio regionų pasiūlymo aibę ir klasifikatoriaus, nustatančio galutinius regionus [64].

Atsižvelgdamas į sudarytus vaizdo požymius, *Faster R-CNN* aptinka objektus taikant dviejų etapų metodą. Pirmajame etape naudojant *RPN* tinklą sukuriamas didelis objektų pasiūlymų rinkinys, nesusietas su jokia klase (angl. *class-agnostics*), kuriame kiekvienas objekto pasiūlymas apibrėžiamas ribojančiu langu *b* ir balu, nusakančiu kaip tikėtina, kad šiame lange yra objektas. Antrajame etape tinklas klasifikuoja kiekvieną pasiūlymą ir numato patikslintą ribojamo lango vietą. Toliau iškarpomos vaizdo požymių dalys, atitinkančios pasiūlymo vaizdo sritį, ir normalizuojamas šios srities dydis. Tinklas naudoja šiuos vaizdo požymius, kad suklasifikuotų ribojantį langą *b* ir objekto klasės balus *s*. Taigi, galiausiai ribojančio lango pasiūlymas yra paverčiamas į regioną *R*. Kiekvienam regionui suteikiamas ribojantis langas *b* ir labiausiai tikėtina klasė *c* kartu su tos klasės patikimumo reikšme *s*. Kaip matoma 2.9 pav., diagramos simboliams eskize yra bandoma nubrėžti kuo labiau atitinkanti rėmelį, pagal tikrąjį objekto dydį. Ši prielaida yra naudojama kituose diagramos skaitmenizavimo žingsniuose. Pavyzdžiui, asociacijos regionų patikslinimo komponente 2.3.2, siekiant teisingai nustatyti asociacijos ryšių pradžios ir pabaigos koordinates arba diagramos simbolių susiejimo komponente 0, susiejant panaudojimo atvejų teksto regioną su panaudojimo atvejų regionu.

2.3.1.1. Priklausomybės ryšio nustatymas

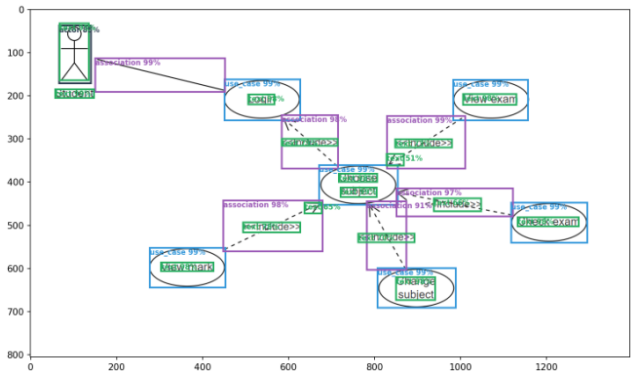
Diagramos simbolių atpažinimo komponentas pateikia keturių skirtingų klasių ribojančius langelius: *association, generalization, use case* ir *actor*. *Association* klasės ribojantis langelis apima asociacijos ir priklausomybės (apėmimo ir išplėtimo) ryšius. Pasirinkta neišskirti atskiros priklausomybės ryšio klasės, nes be papildomo apdorojimo vistiek nebūtų galima nustatyti priklausomybės ryšio tipo: apėmimo ar išplėtimo. Pagal UML 2.5 specifikaciją, priklausomybės ryšiai turi papildomas *<<include>>* ir *<<extend>>* žymas šalia arba ant ryšio vidurio. 2.11 pav. yra pavaizduotas priklausomybės ryšio patikslinimo algoritmas. Apdorojamas kiekvienas *association* klasės regionas

ir tikrinama ar ant ribojančio langelio *mažu* atstumu yra teksto regionas turintis teksto fragmentą *„inc“ (include)* arba *„exc“* (*exclude)*. Jeigu toks teksto regionas yra, *association* ryšys atitinkamai yra paverčiamas į *include* arba *exclude* ryšį. Jeigu ant ribojančo langelio centro yra teksto blokas, tačiau neatitinkantis *include* ir *exclude* žymų, *association* ryšys paverčiamas į *include* tipo ryšį.

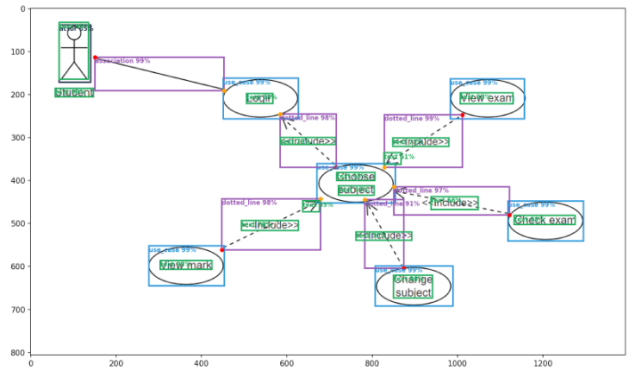


2.11 pav. Priklausomybės ryšio patikslinimo algoritmas

2.12 pav. ir 2.13 pav. yra pateikti pažymėti ribojantys langeliai eskize prieš priklausomybės ryšio patikslinimą ir po. Pradinėje regionų aibėje yra 6 regionai su klase *association,* tačiau po priklausomybės ryšio patikslinimo liko tik 1 *association* ryšio regionas ir 5 *dotted_line* regionai, kurie nusako priklausomybės ryšį.



2.12 pav. Ribojantys langeliai prieš priklausomybės ryšio patikslinimą



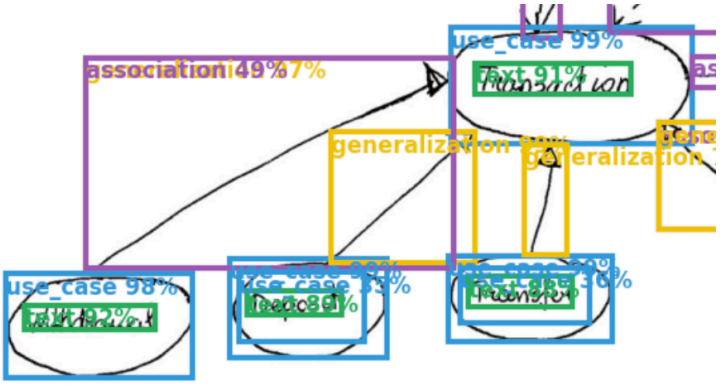
2.13 pav. Ribojantys langeliai po priklausomybės ryšio patikslinimo

Taigi, pasirinkta neišskirti atskiros *priklausomybės* ryšio klasės, nes be papildomo apdorojimo vistiek nebūtų galima nustatyti priklausomybės ryšio tipo: apėmimo ar išplėtimo. Todėl, vykdomas 2.11 pav. pavaizduotas algoritmas priklausomybės ryšio nustatymui iš *association* ryšio klasės.

2.3.1.2. Papildomo apdorojimo žingsniai tikslumui pagerinti

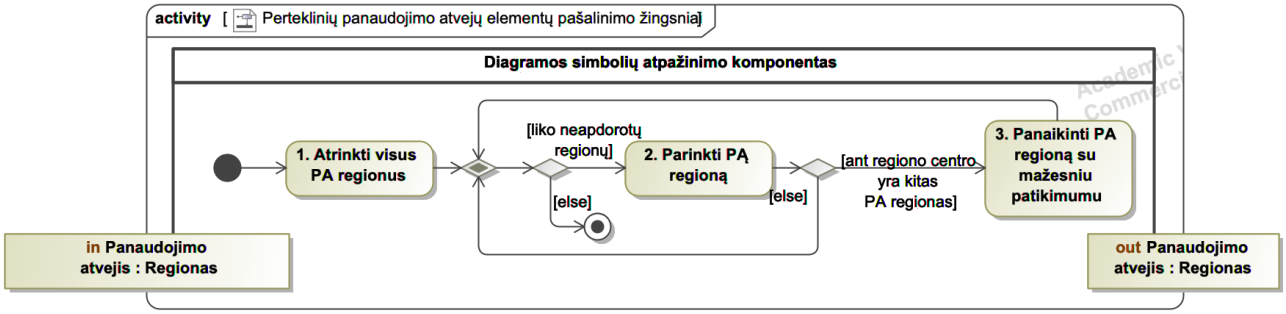
Siekiant patobulinti diagramos simbolių atpažinimo komponento tikslumą yra įgyvendinti papildomo apdorojimo žingsniai: pasikartojančių panaudojimo atvejų pašalinimas bei ryšių, rodančių į tuos pačius elementus, pašalinimas. 2.14 pav. yra pateiktas PA diagramos fragmentas su asociacijos ir paveldėjimo regionais, rodančiais į tuos pačius panaudojimo atvejus; pasikartojančiais panaudojimo atvejų elementais. Pateiktame pavyzdyje, neatlikus papildomo apdorojimo žingsnio, būtų

skaitmenizuoti du pertekliniai panaudojimo atvejai ir vienas neteisingas asociacijos ryšys tarp panaudojimo atvejų, kas lemtų mažesnį galutinį diagramos skaitmenizavimo tikslumą.



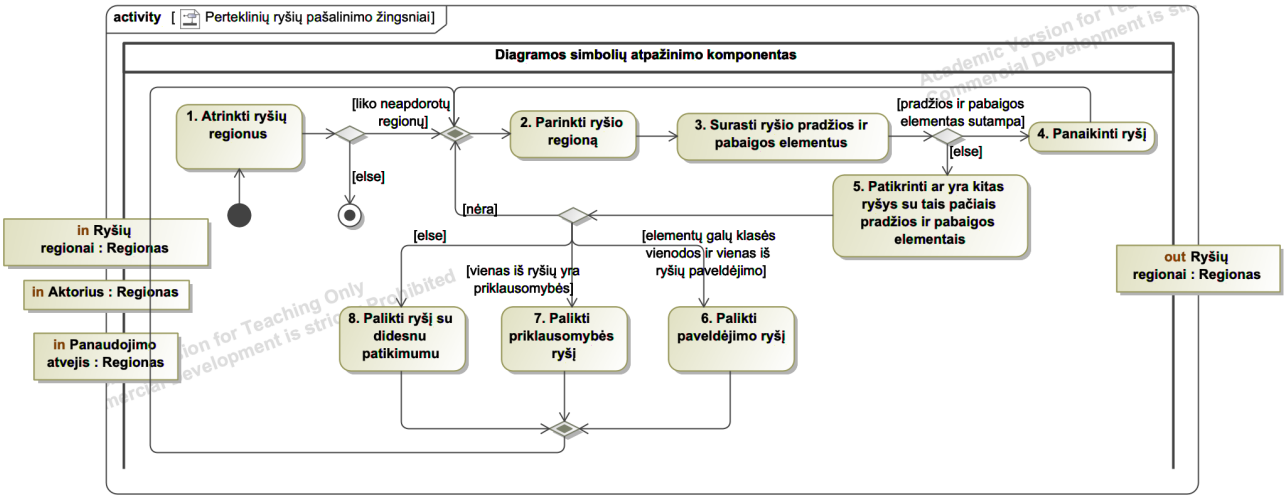
2.14 pav. PA diagramos fragmentas su daugiaklasiu regionu ir pasikartojančiais panaudojimo atvejų elementais.

2.15 pav. yra pateikti perteklinių panaudojimo atvejų elementų pašalinimo algoritmo žingsniai. Pirmiausiai atrenkami visi panaudojimo atvejų regionai iš visų regionų aibės. Patikrinama ar liko neapdorotų regionų. Jeigu liko neapdorotų regionų, parenkamas kitas PA regionas ir patikrinama, ar ant regiono centro yra kitas PA regionas. Tokiu atveju, panaikinamas PA regionas su mažesniu patikimumu. Kitu atveju, apdorojamas kitas regionas iš PA regionų aibės. Algoritmas vykdomas, kol apdorojami visi PA regionai.



2.15 pav. Perteklinių panaudojimo atvejų elementų pašalinimo žingsniai

2.16 pav. yra pateikti perteklinių ryšių pašalinimo algoritmo žingsniai. Pirmiausiai atrenkami ryšių regionai iš visų regionų aibės. Patikrinama ar liko neapdorotų regionų. Jeigu neapdorotų regionų neliko, algoritmas baigiamas. Kitu atveju, parenkamas kitas ryšio regionas ir surandami diagramos elementai esantys ryšio pradžios ir pabaigos galuose (pagal raktinius taškus). Jeigu pradžios ir pabaigos elementas yra tas pats, tai toks ryšys yra panaikinamas iš ryšio regionų aibės. Kitu atveju, patikrinama ar yra kitas ryšys su tais pačiais pradžios ir pabaigos elementais. Jeigu toks ryšys yra, tai tikrinama ar elementų galų klasės vienos, ir vienas iš ryšių yra paveldėjimo. Tokiu atveju paliekamas paveldėjimo ryšys. Jeigu vienas pasikartojančių ryšių yra priklausomybės ryšys, tai jis ir paliekamas. Visais kitais atvejais paliekamas ryšys, turintis didesnį patikimumo balą.

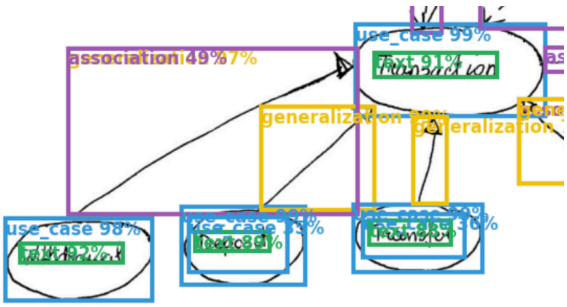


2.16 pav. Perteklinių ryšių pašalinimo žingsniai

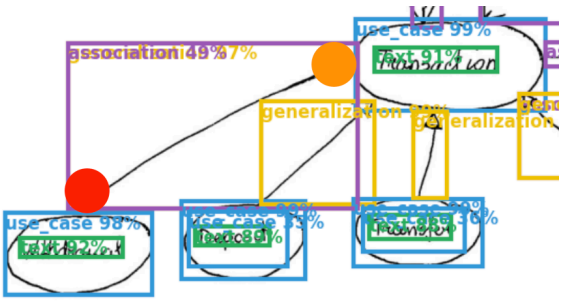
Šiame skyrelyje aprašyti įgyvendinti papildomo apdorojimo žingsniai, pasikartojančių PA regionų (2.15 pav.) ir ryšių pašalinimas (2.16 pav.), kurie leidžia pagerinti galutinės skaitmenizuotos diagramos tikslumą.

2.3.2. Ryšių regionų raktinių taškų nustatymo komponentas

Ryšių regionų raktinių taškų nustatymo komponentas yra reikalingas ryšio regionams *R* nustatyti pradžios (x_{pr}, y_{pr}) ir pabaigos (x_{pb}, y_{pb}) taškų koordinates. Priešingai nei kiti diagramos simboliai, ryšio regionai dažnai turi didelio dydžio ribojantį langą, bet patys simboliai užima tik mažą dalį ribojančio lango ploto. Todėl, ryšio regionams be papildomų pradžios ir pabaigos raktinių taškų negalima teisingai nustatyti, su kuriais kitais diagramos simboliais ryšio regionas turėtų būti sujungtas 0 komponente.



2.17 pav. Paveldėjimo ryšio regiono, besiribojančio su 4 panaudojimo atvejų regionais, be raktinių taškų, pavyzdys.

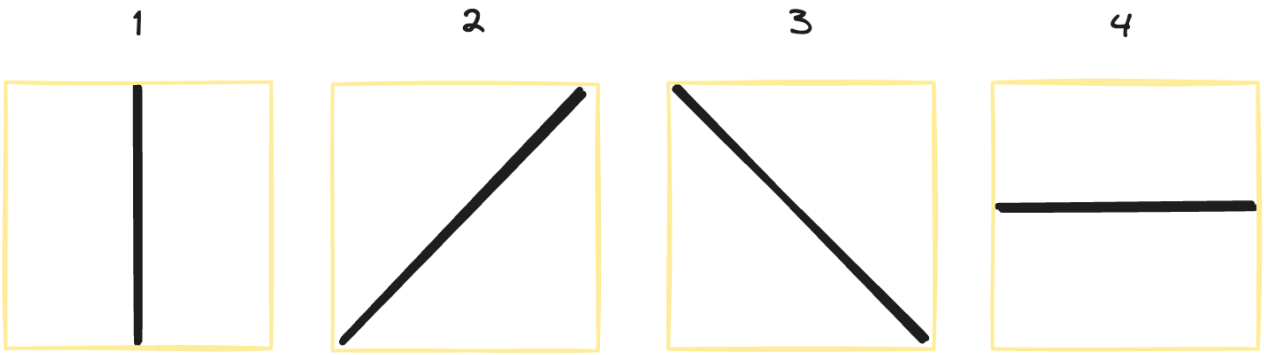


2.18 pav. Paveldėjimo ryšio regiono, besiribojančio su 4 panaudojimo atvejų regionais, su raktiniais taškais, pavyzdys.

2.17 pav. yra pateiktas pavyzdys, kai asociacijos ryšio regionas, ribojasi su keturiais panaudojimo atvejų regionais. Kaip matoma iš paveikslėlio, negalima tiksliai susieti asociacijos ryšio regiono su panaudojimo atvejų regionais „Withdrawal“ ir „Transaction“ neturint (x_{pr}, y_{pr}) ir (x_{pb}, y_{pb}) raktinių taškų. 2.18 pav. pateiktas pavyzdys, kai raktiniai taškai (nuotraukoje raudoni ir oranžiniai taškai) yra nustatyti. Turint ryšio regionio papildomus raktinius taškus, galima nesunkiai susieti ryšio regioną su atitinkamais aktorių arba panaudojimo atvejų regionais.

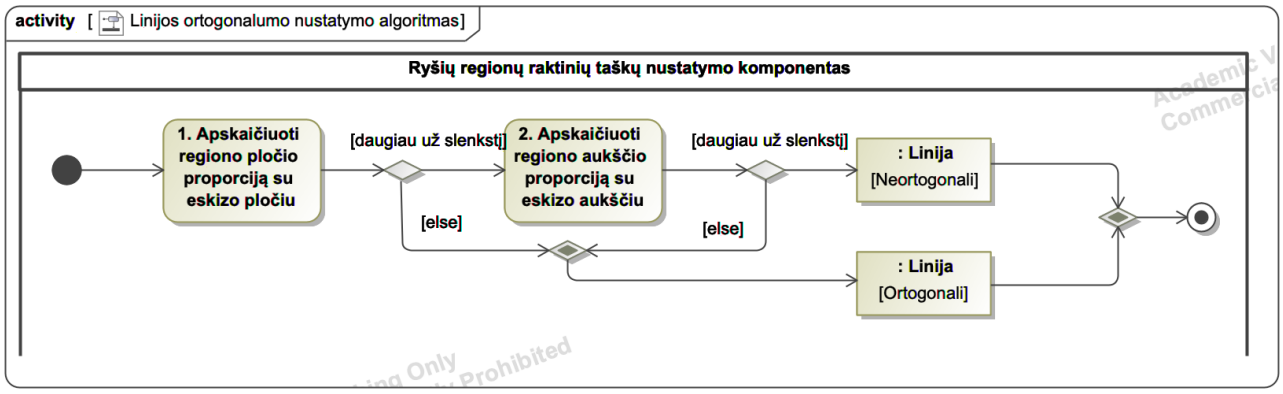
Raktiniai taškai yra nustatomi asociacijos ryšio, apibendrinimo ir apėmimo ryšių regionams *R*. Paprastai asociacijai nėra svarbu, kuris iš taškų yra pradžios ar pabaigos, nes šis ryšys tik nusako, kad simboliai yra *susiję*. Apibendrinimo (angl. *generalization*) ir apėmimo ryšiams (angl. *include* ir *exclude*) yra svarbu teisingai nustatyti, ne tik raktinių taškų koordinates, tačiau ir, kuris iš šių taškų yra pradžios, o kuris pabaigos. Kitu atveju, 2.5.4 komponente, bus neteisingai susieti diagramos simboliai. Pavyzdžiui, neteisingai nustatę, paveldėjimo ryšio pradžios ir pabaigos taškus, tarp dviejų aktorių, bus neteisingai paveldėtas funkcionalumas tarp aktorių.

Šio komponento veikimui yra taikoma prielaida, kad jeigu linija nėra statmena kažkuriai ribojančio langelio kraštinei, tai linija yra išsidėsčiusi per ribojančio langelio įstrižainę, kaip matoma 2.19 pav. *2*, *3* ribojančiuose langeliuose. Linijos *1* ir *4* yra statmenos, todėl jos eina per ribojančio langelio centrą.



2.19 pav. Galimi atpažintų linijų išsidėstymo būdai ir ribojantys rėmeliai

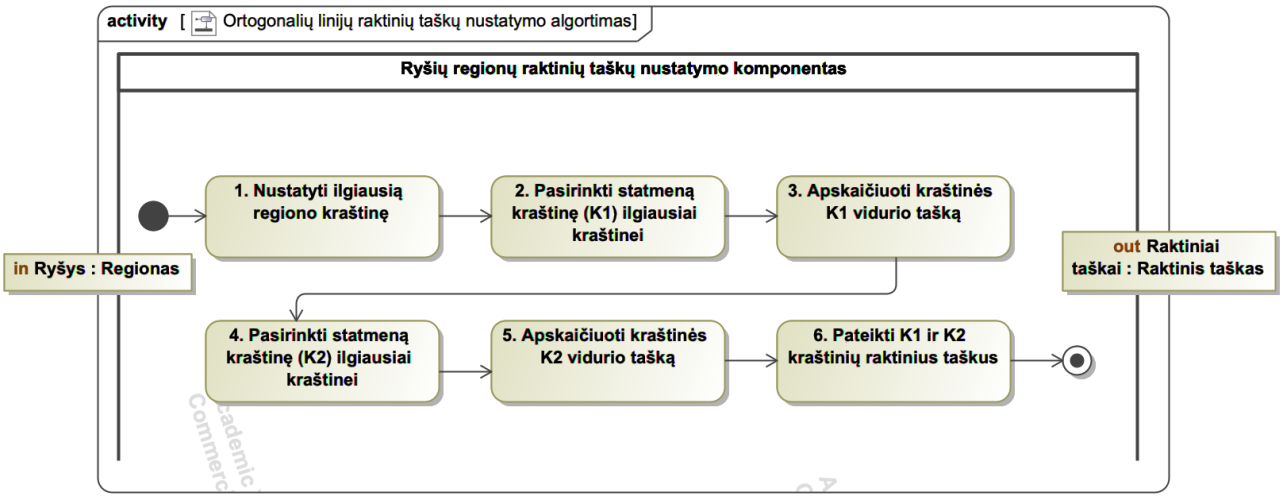
Linijos ortogonalumą su ribojančio langelio kraštine galima nustatyti pagal ribojančio langelio dydį. Linijos ortogonalumo nustatymo algoritmas yra pateiktas 2.20 pav. Jeigu kažkuris iš ribojančio langelio matmenų (plotis arba ilgis) yra mažesnis, negu nustatytas ribinis dydis, laikoma, kad linija yra statmena.



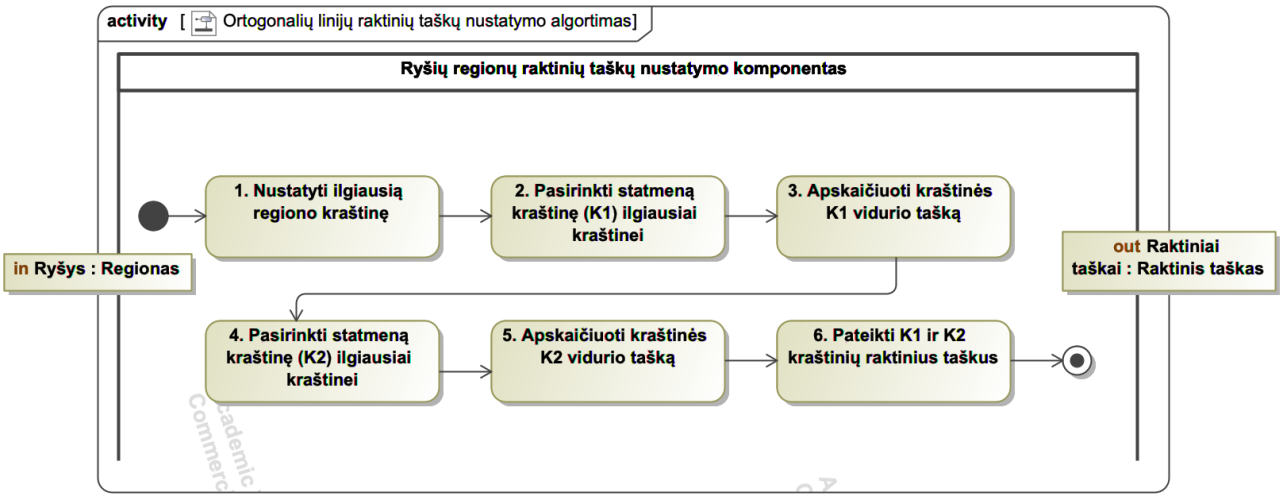
2.20 pav. Linijos ortogonalumo nustatymo algoritmas

Ortogonalioms ir pasvirusioms linijoms taikomi skirtingi algoritmai nustatant pradžios (x_{pr}, y_{pr}) ir pabaigos (x_{pb}, y_{pb}) raktinių taškų koordinates ryšio regionui *R*. Papildomai, paveldėjimo ir apėmimo ryšio regionams nustatomi, kuris iš raktinių taškų yra pradžios, o kuris pabaigos siekiant išlaikyti teisingą diagramos kontekstą tolimesniuose skaitmenizavimo etapuose.

Ortogonalių linijų raktinis taškas (x, y) sudaromas pagal vidurio koordinatės nustatymo formulę: $x_{vid} = \frac{x_1+x_2}{2}$, $y_{vid} = \frac{y_1+y_2}{2}$, kur (x_1, y_1) ir (x_2, y_2) yra regiono *R* kraštinės galų koordinatės. Vidurio koordinatė skaičiuojama trumpajai *R* regiono kraštinei. Taigi, galimos tokios ortogonaliųjų linijų raktinių taškų kombinacijos: (x_{vid}, y_1) ir (x_{vid}, y_2) arba (x_1, y_{vid}) ir (x_2, y_{vid}).

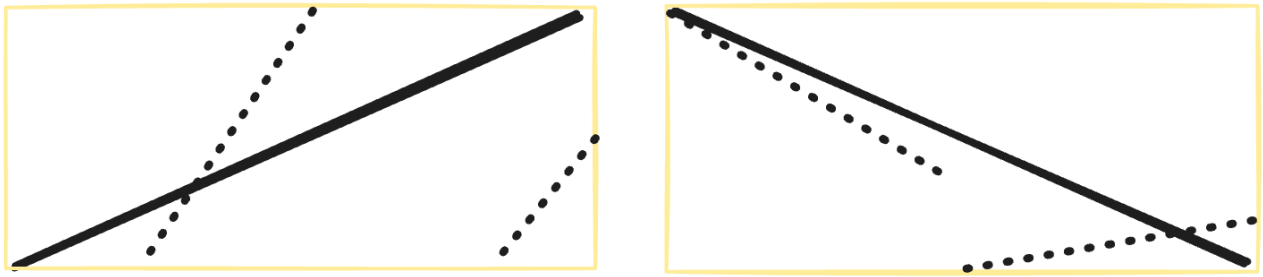


2.21 pav. yra pateiktas ortogonalių linijų raktinių taškų nustatymo algoritmas. Pirmiausia, nustatoma *R* ilgiausia kraštinė. *R* ilgiausiai kraštinei parenkama statmena kraštinė *K1* ir paskaičiuojamas vidurio taškas pagal aukščiau pateiktą formulę. Analogiškai tas pats atliekama su kita kraštine *K2*. Kraštinių *K1* ir *K2* vidurio taškai ir yra linijos raktiniai taškai. Linijos raktinis taškas gali turėti tam tikrą paklaidą, priklausomai nuo ribojančio rėmelio pozicijos, tačiau ši paklaida yra nereikšminga ir neturi įtakos korektiškam diagramos simbolių susiejimui komponente 0.



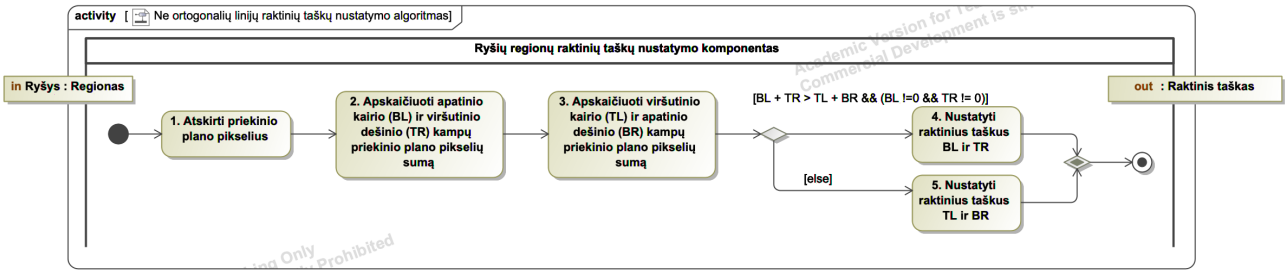
2.21 pav. Ortogonalių linijų raktinių taškų nustatymo algoritmas

Regionams, kuriuose yra neortogonalios linijos, reikia atsižvelgti, kad ribojančiame lange, gali būti persidengusių kitų elementų ar linijų. Tokių linijų raktinių taškų nustatymui neužtenka pasikliauti ribojančio langelio turimais matmenimis. Taikomi morfologinė analizės algoritmai naudojant *OpenCV* biblioteką.



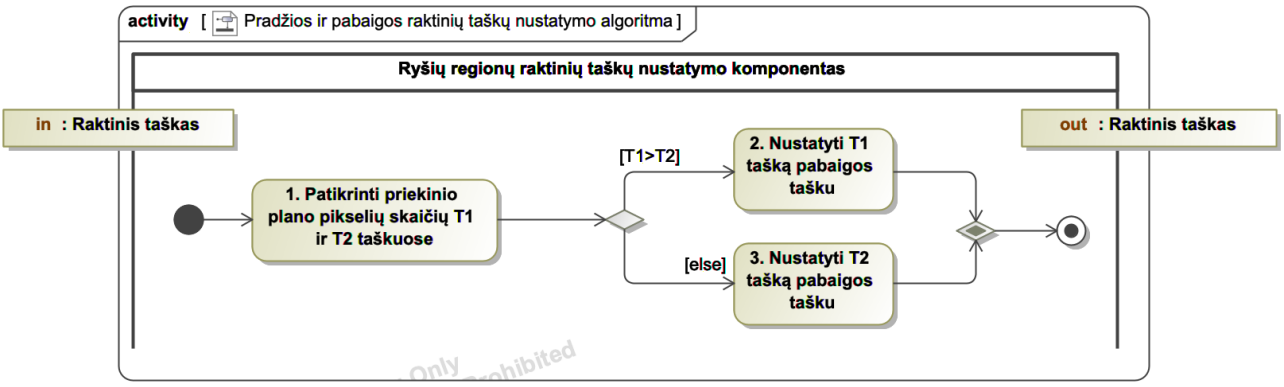
2.22 pav. Linijų pavyzdžiai, kuriuose yra kitų diagramos simbolio elementų

2.23 pav. yra pateiktas supaprastintas ne ortogonalios linijos raktinių taškų nustatymo algoritmas. Pritaikius atitinkamus paveiksliuko apdorojimo morfologinius filtrus, atskiriami priekinio plano (angl. *foreground*) pikseliai nuo fono. Priekinio plano pikseliuose yra diagramos simbolių elementai ir fragmentai. Remiamantis prielaida, kad linijos galai yra regiono *R* kampuose, arba arti kampų, raktiniai regionio taškai yra nustatomi tuose regiono priešinguose kampuose, kur bendra priekinio plano pikselių suma yra didžiausia. Pavyzdžiui, raktiniai taškai bus apatinis kairysis kampas (BL) ir viršutinis dešinysis kampas (BR), jeigu šių kampų priekinio plano pikselių suma bus didesnė už viršutinio kairiojo (TL) ir apatinio dešiniojo (BR) kampo pikselių sumą.



2.23 pav. Ne ortogonalios linijos raktinių taškų nustatymo algoritmas

Norint nustatyti, kuris iš raktinių taškų yra pradžios, o kuris pabaigos taškas, taikomas algoritmas pavaizduotas 2.24 pav. Palyginamas raktinių taškų *T1* ir *T2* priekinio plano pikselių kiekis. Jeigu raktinio taško *T1* pozicijoje yra daugiau priekinio plano pikselių, negu *T2*, tai šis taškas laikomas pabaigos tašku, o taškas *T2* yra nustatomas pradžios tašku. Kitu atveju *T1* yra nustatomas, kaip pradžios taškas, o *T2* pabaigos tašku.

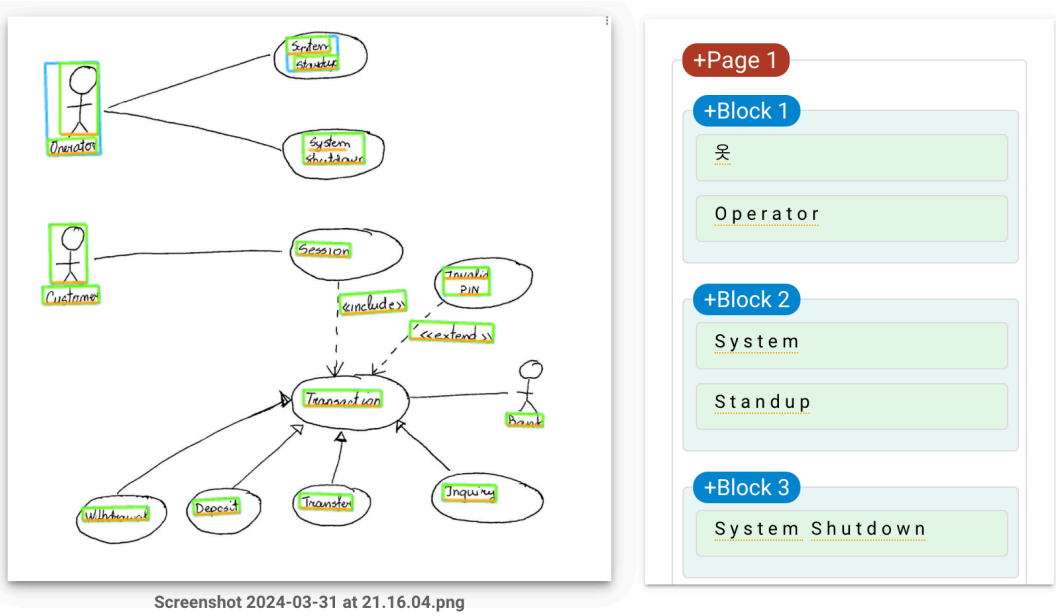


2.24 pav. Pradžios ir pabaigos raktinių taškų nustatymo algoritmas

Apibendrinant, ryšių regionų raktinių taškų nustatymo komponentas yra reikalingas ryšio regionams nustatyti pradžios ir pabaigos raktinių taškų koordinates, be kurių negalima teisingai nustatyti, su kuriais kitais diagramos simboliais ryšio regionas turėtų būti sujungtas. Raktinių taškų algoritmas yra skirtingas, kai ryšys eina per ribojančio langelio įstrižainę ir centrą. Kai ryšys eina per ribojančio langelio centrą – raktiniai taškai yra trumpiausių kraštinių centrai, kitu atveju bandoma surasti, kuriuose ribojančio langelio kampuose raktiniai taškai yra pagal priekinio plano pikselių kiekį. Kitame skyrelyje (2.3.3) aprašomi OCR komponento pagrindiniai veikimo principai.

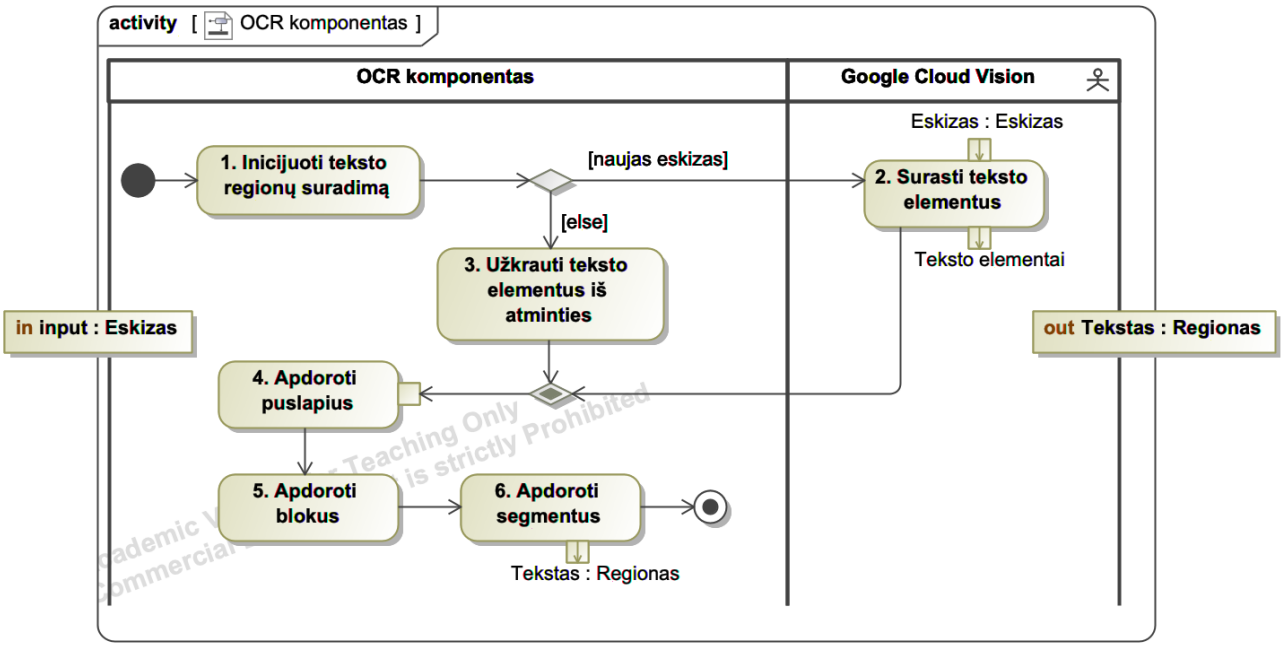
2.3.3. OCR komponentas

OCR komponentas yra reikalingas surasti teksto regionus diagramos eskize. Pateikus eskizą *Google Cloud Vision* paslaugai gaunami teksto regionai išskaidyti pagal hierarchinę struktūrą į puslapius, blokus ir segmentus (2.25 pav.). Puslapis atitinka vieną skaitmenizuojamo eskizo įrašą, blokai atitinka teksto regioną, o segmentai – atskiras linijas bloke.



2.25 pav. Google Cloud Vision OCR teksto blokai eskize

2.26 pav. yra pateiktas OCR komponento veikimo principas. Inicijavus teksto regionų suradimą, patikrinama ar toks eskizas jau buvo apdorotas. Jeigu eskizas nebuvo apdorotas, kreipiamasi į *Google Cloud Vision* paslaugą, kuri grąžina eskizo teksto elementus išskaidytus į puslapius, blokus ir segmentus. Kitu atveju, teksto elementai yra užkraunami iš atminties. Kadangi visada skaitmenizuojama tik po vieną eskizą, gaunamas tik vienas puslapis. Puslapyje esantys blokai turi segmentus, kurie yra išskaidyti per eilutes ir turi būti sujungti. Taigi apdorojus visus teksto elementus išgaunami teksto regionai.

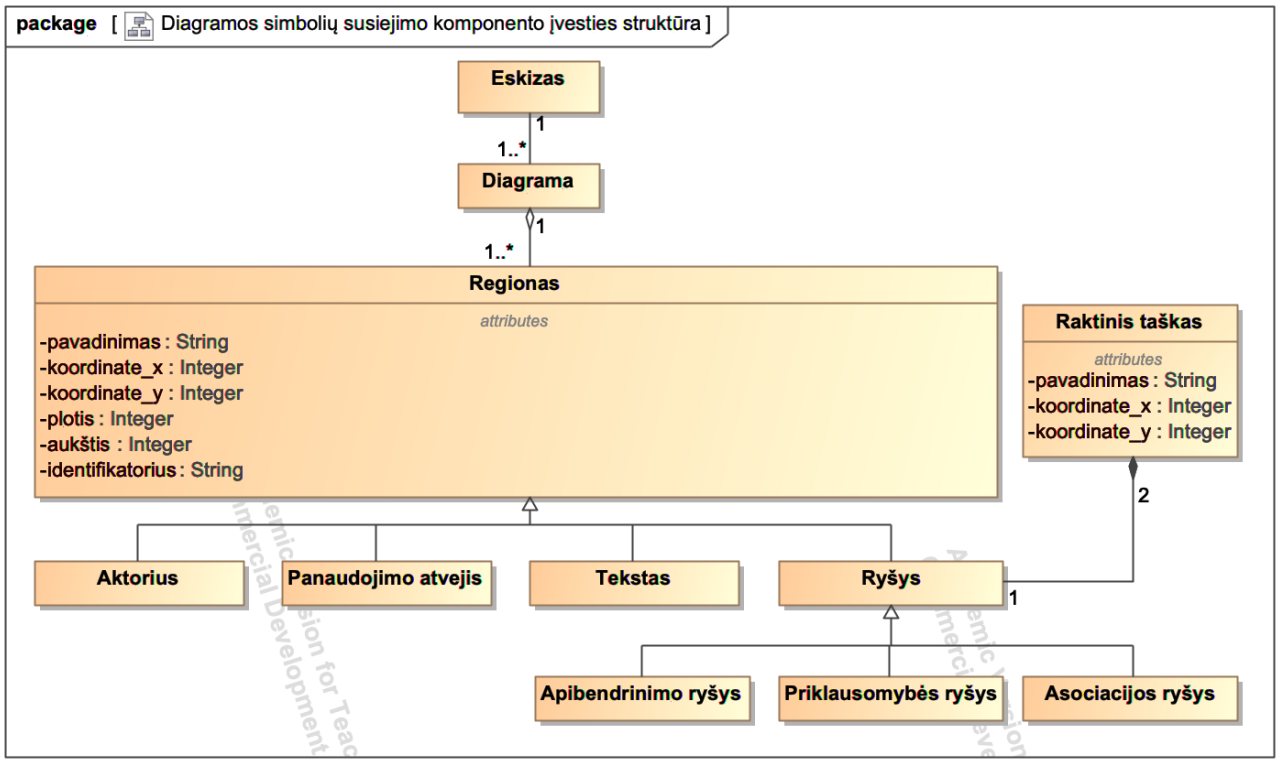


2.26 pav. Teksto regionų suradimo algoritmas OCR komponente

Šiame skyrelyje aprašytas OCR komponentas yra reikalingas surasti teksto regionus diagramos eskize panaudojus *Google Cloud Vision* paslaugą. Toliau aprašomas (2.3.4) diagramos simbolių komponentas, kuris susieja pavienius regionus į bendrą diagramos struktūrą.

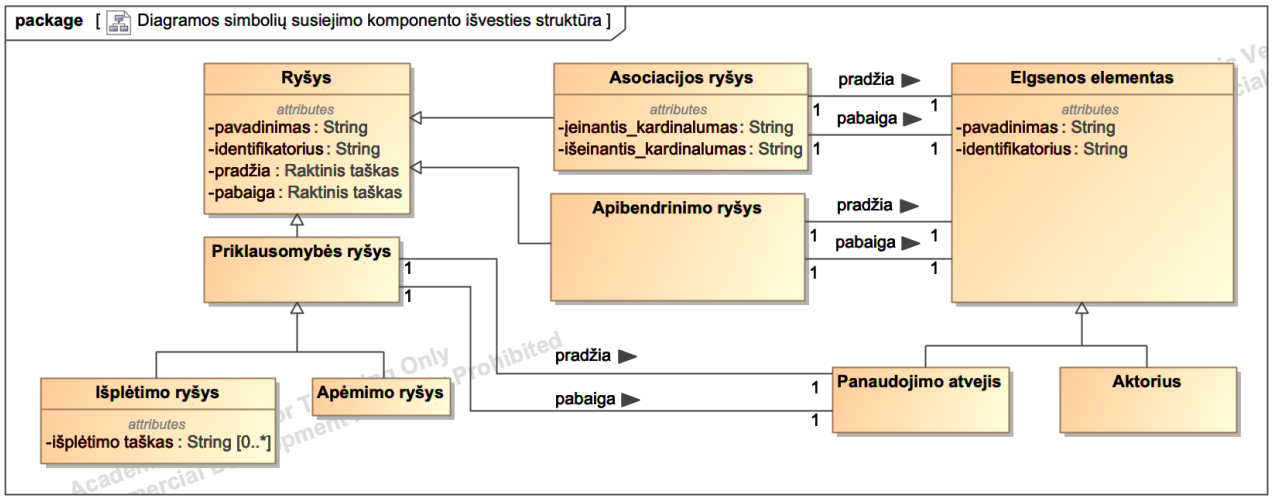
2.3.4. Diagramos simbolių susiejimo komponentas

Diagramos simbolių susiejimo komponento įvestis yra 2.3.1, 2.3.2, 2.3.3 komponentų rezultatas – atskirų regionų informacija JSON formatu. Pagrindinė įvesties struktūra yra pateikta 2.27 pav. diagramoje. Įvestis susideda iš 2.3.1 komponente atpažintų regionų, kur kiekvienas iš jų turi pavadinimą, koordinates x ir y bei plotį ir aukštį. Teksto regionas papildomai turi skaitmenizuotą informaciją apie tekstą, kuris išgaunamas 2.3.3. komponente. Ryšių regionai papildomai turi pradžios ir pabaigos raktinius taškus, kurie nustatomi 2.5.2 komponente.

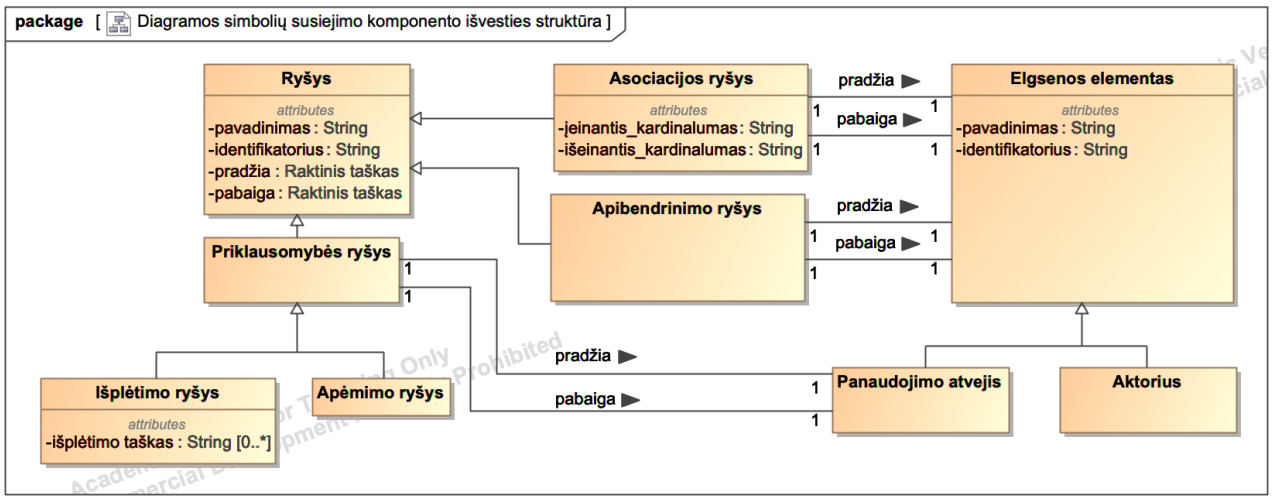


2.27 pav. Diagramos simbolių susiejimo komponento įvesties struktūra

Diagramos simbolių susiejimo komponento išvestis yra JSON formato failas, kuris apibrėžia sujungtus regionus į bendrą visumą – sudarančią UML panaudojimo atvejų diagramą. Išvesties struktūra yra pateikta



diagramoje. Teksto regionai yra suliejami su kitais regionais. Panaudojimo atvejų ir aktoriaus elementui teksto regionas tampa pavadinimu, o asociacijos ryšiams - išplėtimo taškais arba kardinalumu. Panaudojimo atvejų ir aktoriaus elgsenos elementai gali būti susiejami su asociacijos ryšiais. Paveldėjimo ryšiu yra susiejami tik tos pačios klasės elementai. Apėmimo ryšiai siejami tik tarp panaudojimo atvejų.

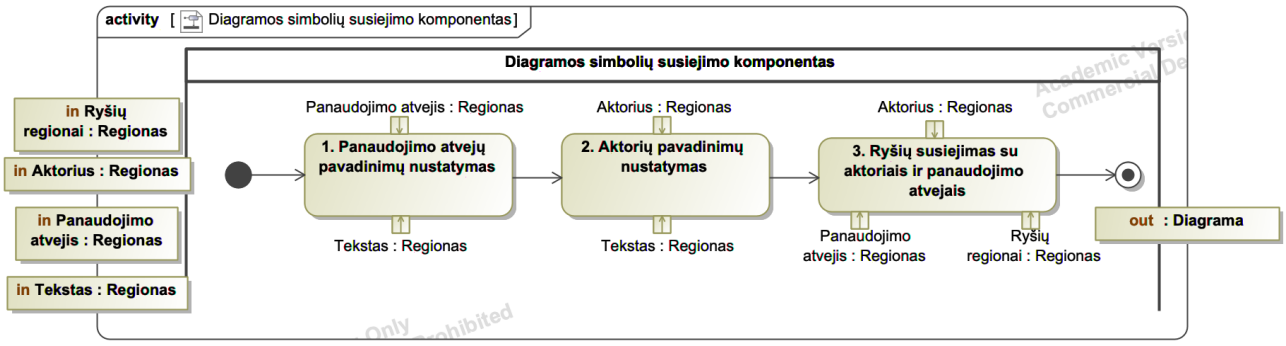


2.28 pav. Diagramos simbolių susiejimo komponento išvesties struktūra

Diagramos simboliai tarpusavyje yra sujungiami remiantis prielaida, kad kuo elementai yra arčiau vienas kito, tuo labiau tikėtina, kad jie yra tarpusavyje susiję. Taigi, elementai tarpusavyje yra susiejami pagal atstumą, kuris yra apskaičiuojamas pagal euklidinio atstumo formulę $d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$, kur p ir q yra taškai koordinačių sistemoje. Taip pat, naudojamasi turimomis žiniomis apie panaudojimo atvejų diagramos sudarymo taisykles:

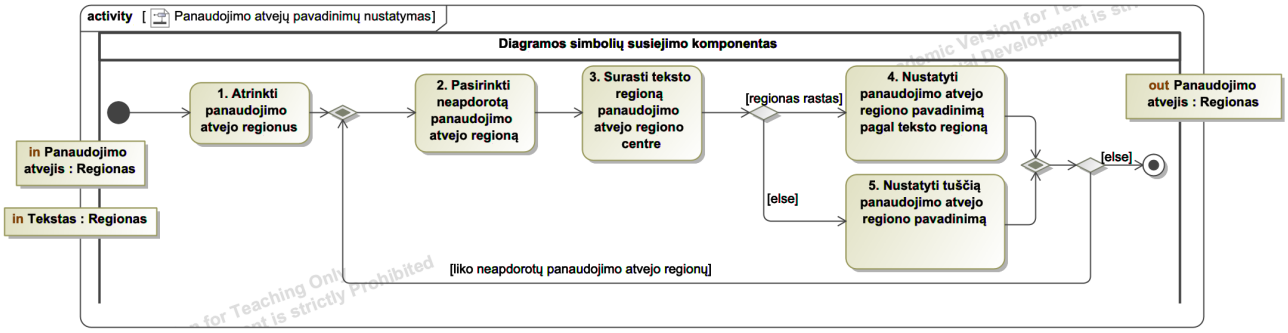
- ryšiai tarpusavyje negali būti susieti;
- apibendrinimo ryšiu gali būti sujungti tik tos pačios klasės elementai;
- priklausomybės ryšiu gali būti sujungti tik panaudojimo atvejai;
- priklausomybės ryšiai turi <<*include*>> arba <<*extend*>> žymas;
- panaudojimo atvejų pavadinimas yra panaudojimo atvejų ribojančio langelio centre;
- aktoriaus pavadinimas dažniausiai būna po aktoriumi;

Diagramos simbolių susiejimo algoritmas yra išskaidomas yra atskirus etapus, kurie pavaizduoti 2.29 pav.. Pirmiausia vykdomas panaudojimo atvejų ir aktorių pavadinimų nustatymas. Tada ryšių regionai susiejami su aktorių ir panaudojimo atvejų regionais.



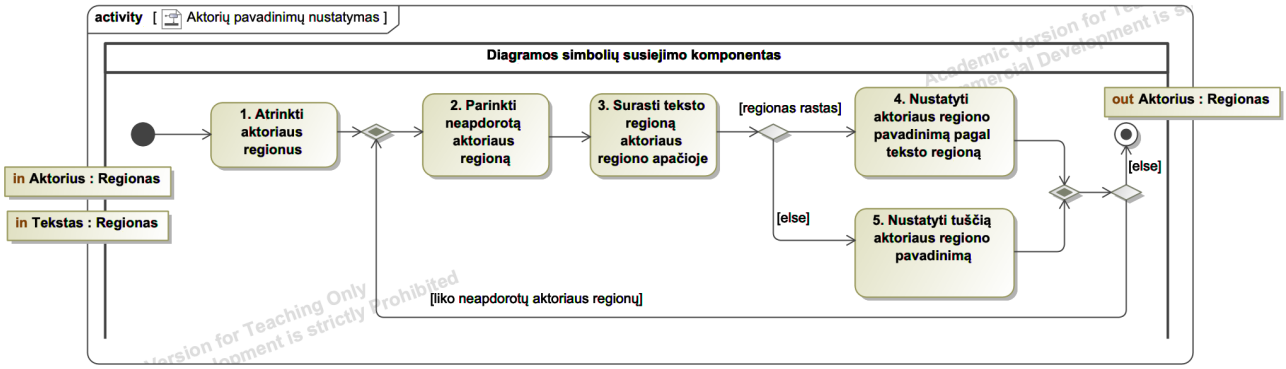
2.29 pav. Pagrindiniai diagramos simbolių susiejimo algoritmo žingsniai

2.30 pav. yra pateiktas panaudojimo atvejų pavadinimo nustatymo algoritmas. Apdorojami visi panaudojimo atvejų regionai. Tikrinama ar panaudojimo atvejų regiono centre yra teksto regionas. Jeigu teksto regionas yra, tada panaudojimo atvejo pavadinimas yra nustatomas pagal teksto regiono turinį. Kitu atveju, panaudojimo atvejo regionas yra paliekamas tuščias.



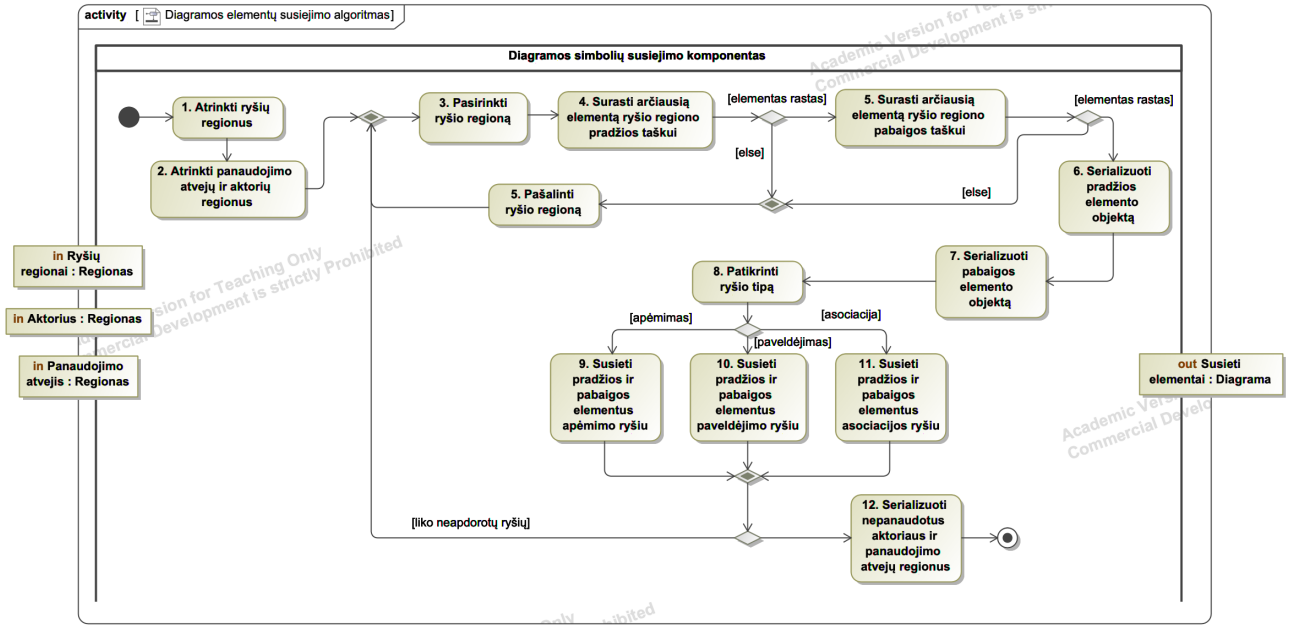
2.30 pav. Panaudojimo atvejų pavadinimo nustatymo algoritmas

2.31 pav. yra pateiktas aktorių pavadinimų nustatymo algoritmas. Apdorojami visi aktoriaus regionai. Tikrinama ar aktoriaus regiono apačioje yra teksto regionas. Jeigu teksto regionas yra, tada aktoriaus pavadinimas yra nustatomas pagal teksto regiono turinį, kitu atveju – pavadinimas paliekamas tuščias.



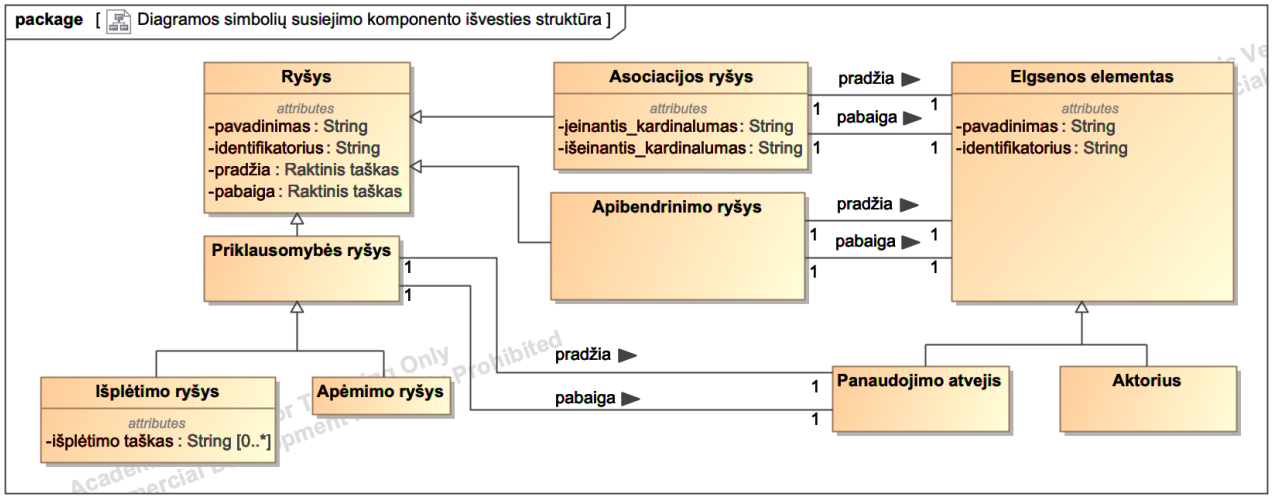
2.31 pav. Aktorių pavadinimo nustatymo algoritmas

2.32 pav. yra pateiktas diagramos elementų susiejimo supaprastintas algoritmas. Iš visų regionų aibės atrenkami ryšių, panaudojimo atvejų ir aktorių regionai. Pasirenkamas ryšio regionas, kuriam bandoma surasti arčiausius elementus (aktorių arba panaudojimo atvejį) pradžios ir pabaigos raktiniams taškams. Jeigu bent kuriam iš raktinių taškų nepavyko surasti elemento, toks ryšys yra pašalinamas iš ryšių regionų aibės. Kitu atveju, serializuojami surasti pradžios ir pabaigos elementai. Toliau šie elementai yra atitinkamai susiejami su paveldėjimo, apėmimo arba paprastu asociacijos ryšiu. Šis procesas tęsiamas, kol apdorojami visi ryšio regionai. Galiausiai serializuojami visi nepanaudoti aktoriaus ir panaudojimo atvejų regionai, kurie diagramoje nebus sujungti su jokiu ryšiu.



2.32 pav. Diagramos elementų susiejimo algoritmas

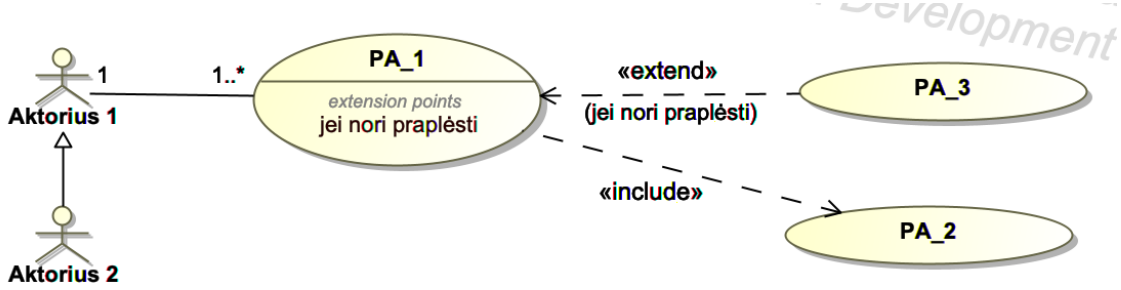
Atlikus visus diagramos simbolių susiejimo komponento algoritmo žingsnius, sudaroma



pavaizduota struktūra, iš kurios galima sugeneruoti XMI failą. Pastarąjį toliau galima įkelti į tokį formatą palaikantį CASE įrankį ir pratęsti diagramos eskizo modeliavimą.

2.3.5. Diagramos konvertavimo į XMI failą komponentas

Diagramos konvertavimo į XMI failą komponento įvestis yra susieta diagramos struktūra 2.28 pav. JSON formatu, o išvestis - XMI failas. Diagramos serializacija į XMI failą yra atliekama pagal XMI specifikaciją [14].



2.33 pav. Panaudojimo atvejų diagramos pavyzdys

```
<xmi:XMI xmlns:uml="http://www.omg.org/spec/UML/20110701" xmlns:xmi=" http://www.omg.org/spec/UML/20110701">
<uml:Package xmi:id="pa" xmi:label="PA" name="Generated UC Diagram">
    <packagedElement xmi:type='uml:Actor' xmi:id='actor_1' name='Aktorius 1'/>
    <packagedElement xmi:type='uml:UseCase' xmi:id='use_case_1' name='PA_1'>
        <extensionPoint xmi:type='uml:ExtensionPoint' xmi:id='extension_point_1' name='jei nori praplėsti' visibility='public'>
            <extension xmi:idref='extend_1'/>
        </extensionPoint>
        <include xmi:type='uml:Include' xmi:id='include_1' visibility='public' addition='use_case_2'/>
    </packagedElement>
    <packagedElement xmi:type='uml:Association' xmi:id='association_1'>
        <memberEnd xmi:idref='association_1_member_end_1'/>
        <memberEnd xmi:idref='association_1_member_end_2'/>
        <navigableOwnedEnd xmi:idref='association_1_member_end_1'/>
        <navigableOwnedEnd xmi:idref='association_1_member_end_2'/>
        <ownedEnd xmi:type='uml:Property' xmi:id='association_1_member_end_1' visibility='private' type='use_case_1' association='association_1'>
            <xmi:Extension extender='MagicDraw UML 2022x'>
                <modelExtension>
                    <lowerValue xmi:type='uml:LiteralInteger' xmi:id='association_1_member_end_1_c_lower' value='1'/>
                </modelExtension>
            </xmi:Extension>
            <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='association_1_member_end_1_c_upper' value='*'/>
        </ownedEnd>
        <ownedEnd xmi:type='uml:Property' xmi:id='association_1_member_end_2' visibility='private' type='actor_1' association='association_1'>
            <xmi:Extension extender='MagicDraw UML 2022x'>
                <modelExtension>
                    <lowerValue xmi:type='uml:LiteralInteger' xmi:id='association_1_member_end_2_c_lower' value='1'/>
                </modelExtension>
            </xmi:Extension>
            <xmi:Extension extender='MagicDraw UML 2022x'>
                <modelExtension>
                    <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='association_1_member_end_2_c_upper' value='1'/>
                </modelExtension>
            </xmi:Extension>
        </ownedEnd>
    </packagedElement>
    <packagedElement xmi:type='uml:Actor' xmi:id='actor_2' name='Aktorius 2'>
        <generalization xmi:type='uml:Generalization' xmi:id='generalization_1' general='actor_1'/>
    </packagedElement>
    <packagedElement xmi:type='uml:UseCase' xmi:id='use_case_2' name='PA_2'/>
    <packagedElement xmi:type='uml:UseCase' xmi:id='use_case_3' name='PA_3'>
        <extend xmi:type='uml:Extend' xmi:id='extend_1' visibility='public' extendedCase='use_case_1'>
            <extensionLocation xmi:idref='extension_point_1'/>
        </extend>
    </packagedElement>
</uml:Package>
</xmi:XMI>
```

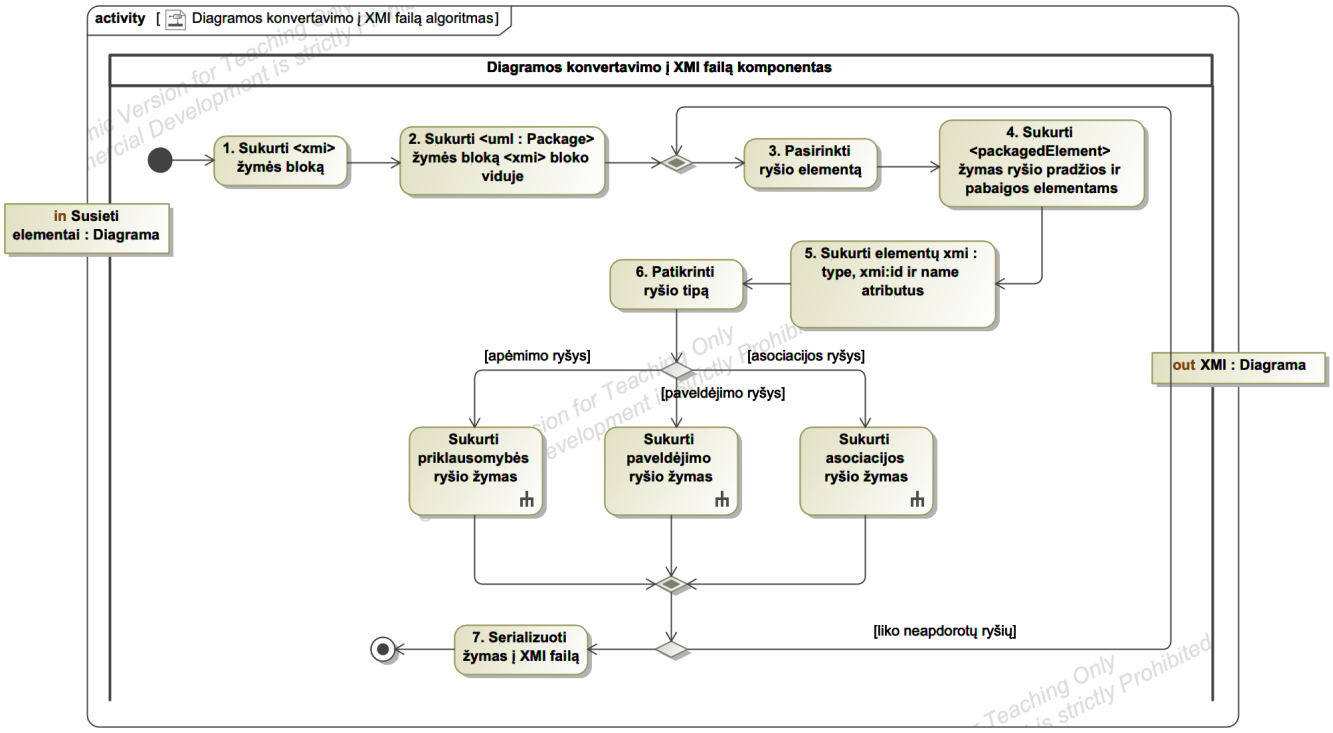
2.34 pav. Panaudojimo atvejų pavyzdžio diagrama XMI formatu.

2.34 pav. yra XMI failas atitinkantis 2.33 pav. pavaizduotą panaudojimo atvejų diagramą. Pateiktame XMI failo pavyzdyje matomos visos reikalingos XML žymos, norint sukurti paketą, panaudojimo diagramos simbolius, bazines jų savybes bei susieti simbolius tarpusavyje ryšiais.

XMI failas pradedamas kurti su žyme ***<xmi:XMI>***. Sukuriamas paketas su žyme ***<uml:Package>***, kuriame talpinami skaitmenizuotos diagramos simboliai. Žymės atributais ***xmi:id*** ir ***name*** nurodo paketo identifikatorių ir pavadinimą. Toliau pakete kuriami skaitmenizuoti diagramos simboliai su žyme ***<packagedElement>***. Kiekviena simbolio žymė turi atributus ***xmi:type***, ***name*** ir ***xmi:id***, kuris naudojamas susieti elementus tarpusavyje su žyme ***xmi:idref***, nurodant norimo susieti elemento identifikatorių. Asociacijos tipo simboliai savyje turi žymes ***<memberEnd>***, ***<navigableOwnedEnd>*** ir ***<ownedEnd>***, kuriomis ryšių galams priskiriami diagramos simboliai pagal jų identifikatorius. ***<ownedEnd>*** žymos bloke galima sukurti ***<lowerValue>*** ir ***<upperValue>*** žymas, kuriomis

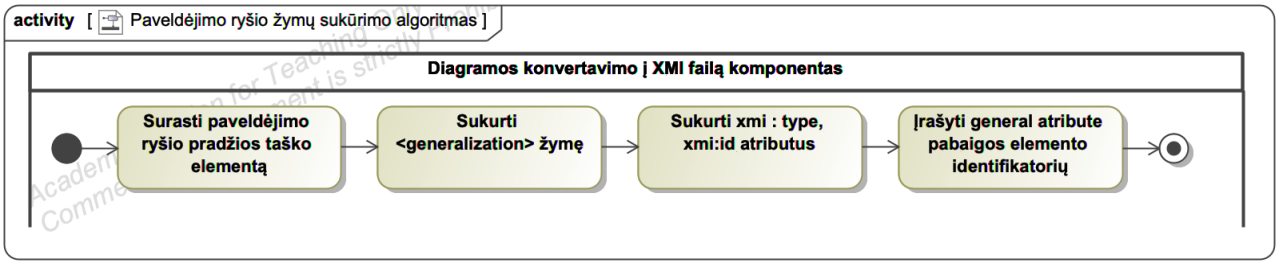
nusakomas ryšio galo kardinalumas. Norint pažymėti paveldimumo ryšį naudojama ***<generalization>*** žyma elemento viduje, kuris paveldi norimo simbolio, pažymėto ***general*** atributu, savybes. *Include* apėmimo ryšys yra valdomas ***<include>*** žyma, o *extend* išplėtimo ryšys ir praplėtimo taškai yra sukuriami žymomis: ***<extend>***, ***<extension>***, ***<extensionPoint>*** ir ***<extensionLocation>***.

2.35 pav. yra pateiktas diagramos konvertavimo į XMI failą komponento algoritmas. Sukuriama ***<xmi>*** žymė ir jos viduje ***<uml:Package>*** žymės blokas. Toliau apdorojamas kiekvienas ryšio elementas. Sukuriamos ***<packagedElement>*** žymos ryšio pradžios ir pabaigos elementams. Žymos užpildomos ***xmi:type***, ***xmi:id*** ir ***name*** atributais. Patikrinamas ryšio tipas ir atitinkamai sukuriamos reikalingos priklausomybės ryšio, paveldėjimo ryšio arba asociacijos ryšio žymos. Galiausiai visos žymos serializuojamos į XMI failą.



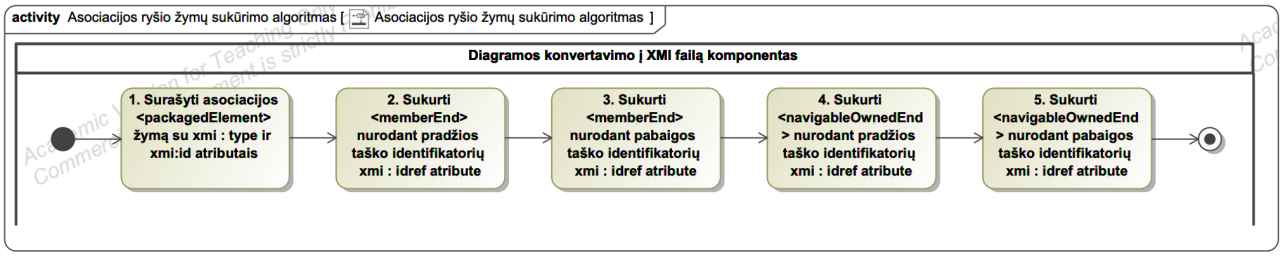
2.35 pav. Diagramos konvertavimo į XMI failą komponento algoritmas

2.36 pav. yra pavaizduotas paveldėjimo ryšio žymų sukūrimo algoritmas. Pirmiausia surandamas paveldėjimo ryšio pradžios taške esantis elementas. Elemento žymos bloko viduje sukuriama ***<generalization>*** žyma su ***xmi:type*** ir ***xmi:id*** atributais. Atributu ***general*** yra nustatomas ryšio pabaigos taško simbolio identifikatorius.



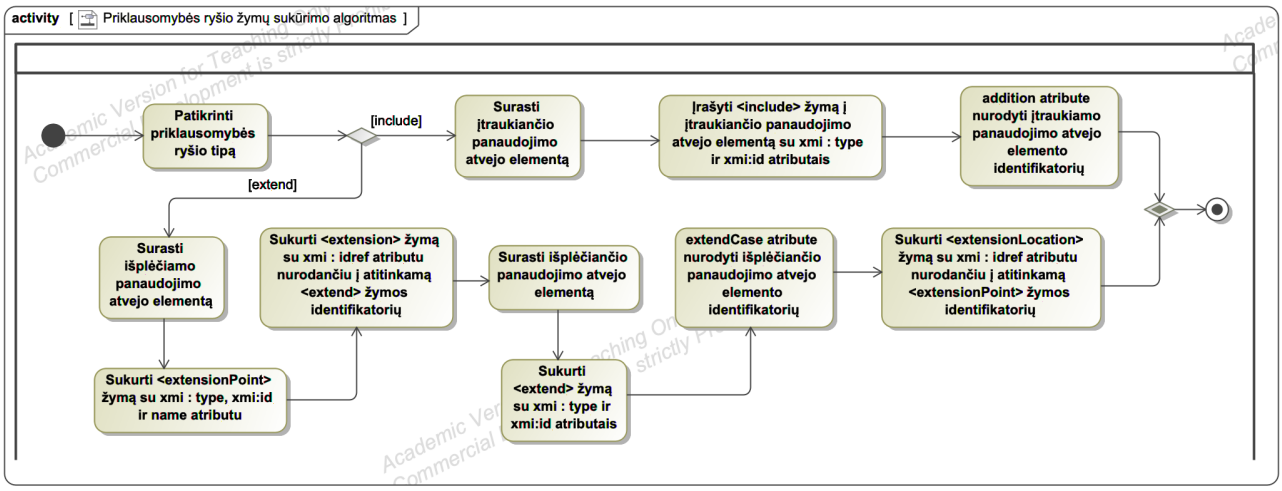
2.36 pav. Sukurti paveldėjimo ryšio žymas algoritmas

2.37 pav. yra pavaizduotas asociacijos ryšio žymų sukūrimo algoritmas. Pirmiausia sukuriama asociacijos elemento žymą ***<packagedElement>*** su ***xmi:id*** ir ***xmi:type*** atributais. Galiausiai sukuriamos ***<memberEnd>*** ir ***<navigableEnd>*** žymos kuriuose nurodomo ryšio pradžios ir pabaigos elementų identifikatoriai ***xmi:idref*** atributuose.



2.37 pav. Sukurti asociacijos ryšio žymas algoritmas

2.38 pav. yra pavaizduotas priklausomybės ryšio žymų sukūrimo algoritmas. Pirmiausia patikrinamas priklausomybės ryšio tipas. Jeigu tai apėmimo ryšys, tada surandamas įtraukiančio panaudojimo atvejų elementas ir sukuriama ***<include>*** žyma su ***xmi:type*** ir ***xmi:id*** atributais. **Addition** atribute nurodomas įtraukiamo panaudojimo atvejo identifikatorius. Jeigu tai išplėtimo ryšys, tada surandamas išplečiamo panaudojimo atvejo elementas. Sukuriama ***<extensionPoint>*** žyma su ***xmi:type***, ***xmi:id*** ir ***name*** atributais. Žymos viduje sukuriama ***<extension>*** žyma su ***xmi:idref*** atributu, nurodančiu į atitinkamą ***<extend>*** žymos identifikatorių. Toliau surandamas išplėčiančio panaudojimo atvejo elementas. Elemento viduje sukuriama ***<extend>*** žyma su ***xmi:type*** ir ***xmi:id*** atributais. ***ExtendCase*** atribute nurodomas išplečiančio panaudojimo atvejo elemento identifikatorių. Galiausiai sukuriama ***<extensionLocation>*** žyma su ***xmi:idref*** atributu nurodančiu į atitinkamą ***<extensionLocation>*** žymos identifikatorių.



2.38 pav. Sukurti priklausomybės ryšio žymas algoritmas

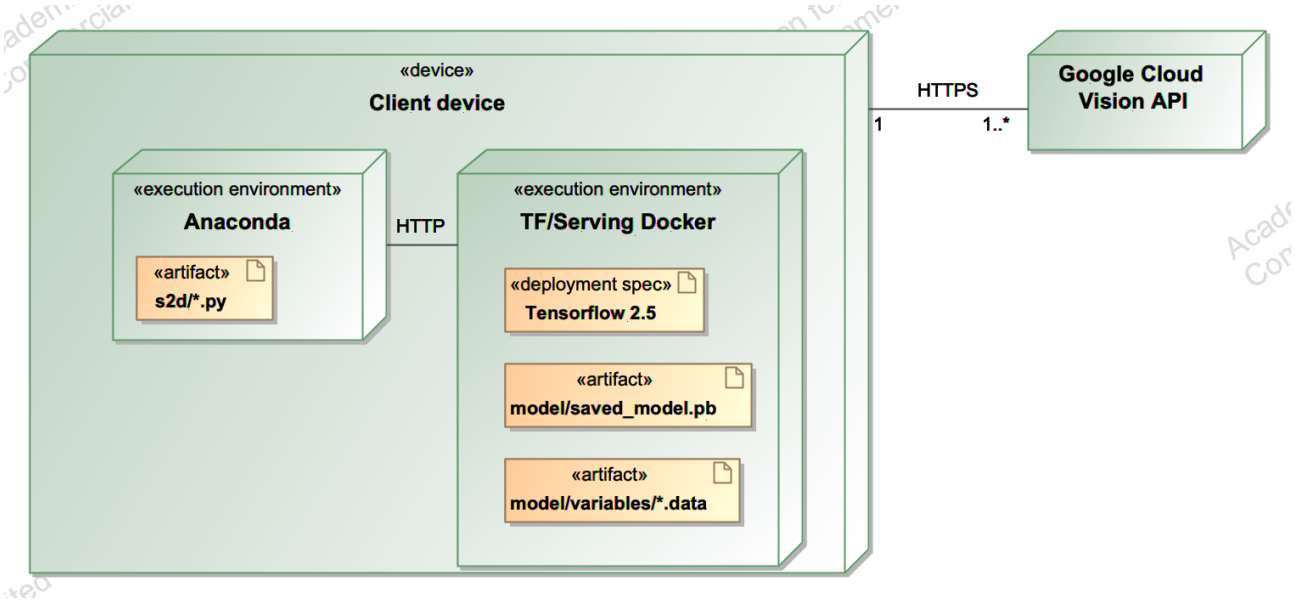
Taigi, atlikus visus komponento algoritmo veiksmus gaunamas XMI failas, savo turiniu panašus į pateiktą 2.34 pav. pavyzdį. Toliau naudotojas gali XMI failą atsidaryti XMI formatą palaikantį įrankį įrankyje, pavyzdžiui, *MagicDraw* ir pratęsti modeliavimą.

3. Sprendimo realizacijos projektas

Šiame skyriuje yra aprašytas sprendimo realizacijos projektas. 3.1 skyrelyje aprašytas statinis sistemos vaizdas išreikštas diegimo, komponentų ir paketų diagramose. 3.2 skyrelyje aprašytas panaudojimo atvejų diagramų eskizų duomenų rinkinys ir duomenų rinkinio pagrindinės charakteristikos. 3.3 skyrelyje aprašytas diagramos simbolių atpažinimo modelio pasirinkimas ir apmokinimas. 3.4 skyrelyje pateikta sistemos rankinio testavimo eiga.

3.1. Statinis sistemos vaizdas

3.1 pav. yra pavaizduota „S2D“ (angl. *Sketch To Diagram*) UML eskizų skaitmenizavimo sistemos diegimo diagrama. Sistema yra monolitinės architektūros – naudotojo sąsaja bei serverio dalis, yra talpinami *s2d/*.py* artefaktuose. Sistemos paleidimui reikalinga įsirašyti įvairias bibliotekas, todėl jas įsidiegti rekomenduojama *Anaconda* virtualioje *Python* vykdymo aplinkoje. Sistemos veikimui papildomai reikia autentikuotis su komandinės eilutės įrankiu *gcloud,* kuriuo gaunama prieiga prie *Google Cloud Vision API* paslaugos. Tam pirmiausia reikalinga susikurti *Google Cloud Vision* projektą *Google* pagrindinėje konsolėje. Toliau reikia paleisti *TF/Serving Docker* konteinerio paslaugą, kuri per *HTTPS* protokolą suteikia galimybę naudotis sukurtu UML diagramos simbolių atpažinimo modeliu. Tokiu būdu yra supaprastinamas „S2D“ sistemos įdiegimas naudotojui, nes kitu atveju reikėtų įsidiegti *TensorFlow 2.5* aplinką kiekvienam naudotojui atskirai. Sistemos kūrimo metu buvo apsvarstyta galimybė atlikti naudotojo sąsajos ir serverio diegimą atskiruose serveriuose, tačiau dėl projekto dydžio buvo nuspręsta sistemą kurti monolitinės architektūros pagrindu. Monolitines sistemas paprasčiau testuoti, kurti, prižiūrėti, pasileisti ir diegti, nei mikroserviso architektūros sistemas. Galiausiai pradinėje projekto fazėje monolitinė architektūra leidžia sutelkti dėmesį į pagrindinių funkcijų kūrimą, nesusiduriant su paskirstytų sistemų sudėtingumu ir iššūkiais.



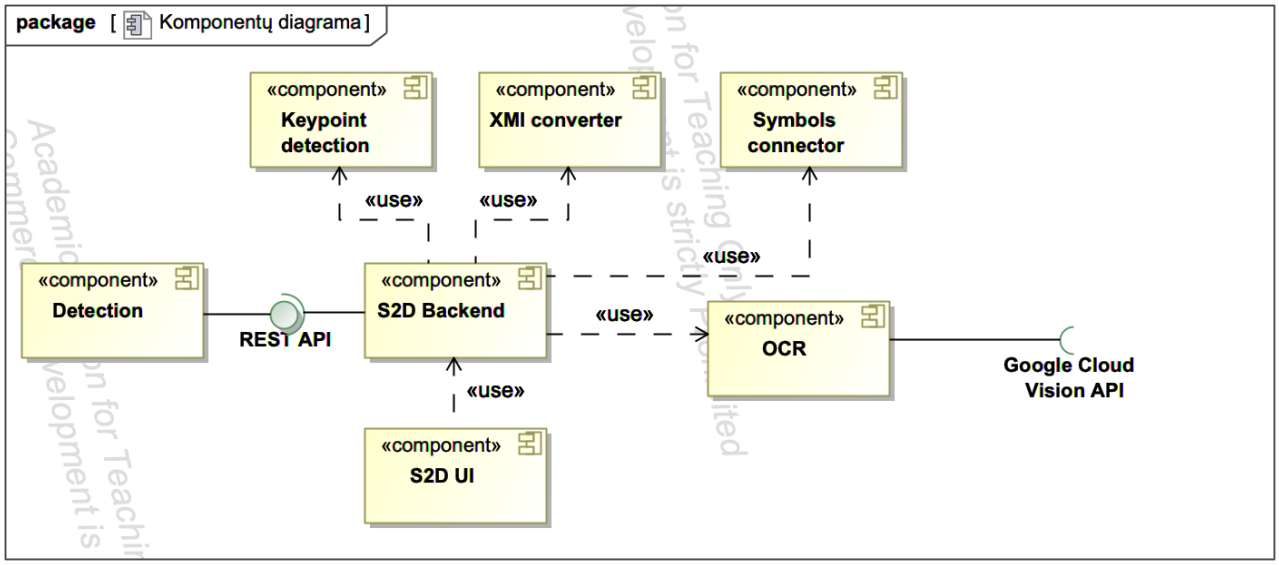
3.1 pav. „S2D“ sistemos diegimo diagrama

3.2 pav. yra pavaizduota UML diagramų eskizų skaitmenizavimo sistemos komponentų diagrama. Pagrindiniai sistemos loginiai komponentai yra *S2D UI* ir *S2D Backend*.

– *S2D Backend* komponentas naudoja *Detection*, *OCR*, *Keypoint detection*, *XMI converter* ir *Symbols connector* komponentus. *Detection* komponentas teikia mašininio modelio UML

simbolių atpažinimo REST API paslaugą per HTTP protokolą. *OCR* komponentas komunikuoja su *Google Cloud Vision API* ir yra naudojamas teksto išgavimui iš paveiksliukų. *XMI converter* komponentas yra naudojamas skaitmenizuotos diagramos konvertavimui į XMI formato failą. *Keypoint detection* komponentas yra atsakingas už ryšių regionų raktinių taškų nustatymą. *Symbols connector* komponentas sujungia atskirus diagramos simbolius;

– *S2D UI* komponentas realizuoja S2D naudotojo sąsają. Šis komponentas naudoja *S2D Backend* komponentą iškviečiant UML panaudojimo atvejų diagramos skaitmenizavimo ir konvertavimo į XMI failą paslaugas.



3.2 pav. „S2D“ sistemos komponentų diagrama

Taigi, 3.1 poskyryje aprašytas statinis sistemos vaizdas pateikiant diegimo ir komponentų diagramas. Kitame poskyryje 3.2 aprašomas panaudojimo atvejų diagramų eskizų duomenų rinkinio sudarymas ir pagrindinės duomenų rinkinio charakteristikos.

3.2. Duomenų rinkinio sudarymas

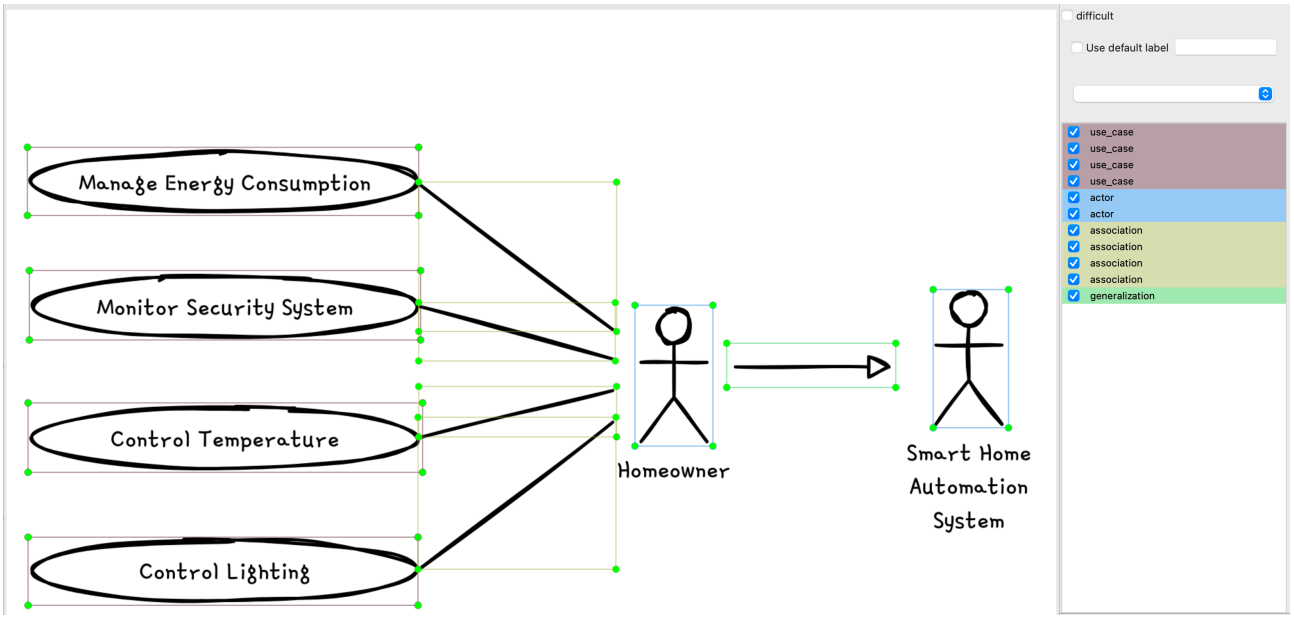
Diagramos simbolių atpažinimo komponentui realizuoti (2.3.1) buvo sudarytas naujas panaudojimo atvejų diagramų rinkinys. Šiame rinkinyje yra 346 diagramų eskizai, iš kurių 288 (83.2%) yra diagramos nubraižytos su įrankiu, o 58 (16.7%) diagramos nubraižytos ranka. Kuriant duomenų rinkinį buvo kuriami įvairių dydžių elementai įvairiomis padėtimis, siekiant pasiekti geresnį modelio tikslumą. Iš viso diagramas sudarė 3 skirtingi asmenys. Suanotuoto duomenų rinkinio statistika pateikiama 3.1 lentelėje. Asociacijos ryšio elementų duomenų rinkinyje yra 1907 (36.6%), panaudojimo atvejų elementų – 1803 (34.6%), aktoriaus elementų – 838 (16.1%), paveldėjimo ryšio elementų – 658 (12.7%). Asociacijos ryšio elementus sudaro paprasti asociacijos ryšiai ir priklausomybės ryšiai. Duomenų rinkinys nėra tolygiai pasiskirstęs, bet ir nėra tokių klasių, kurias būtų dėl to sudėtinga atpažinti. Verta paminėti, kad anotuojant duomenų rinkinį buvo sužymėti ir 3225 teksto regionai, tačiau apmokinimo metu nebuvo panaudoti, nes teksto elementų informacijai surasti naudojama išorinė *Google Cloud Vision OCR* paslauga (2.3.3).

3.1 lentelė. PA diagramų duomenų rinkinio pasiskirstymas per klasę

Klasė	Kiekis	Procentinė išraiška

Asociacijos ryšys	1907	36.6%
Panaudojimo atvejis	1803	34.6%
Aktorius	838	16.1%
Paveldėjimo ryšys	658	12.7%

Vieną duomenų rinkinio eilutę sudaro eskizo paveikslėlis *.png* arba *.jpg* formatu ir informacija apie diagramoje esančių elementų koordinates ir kitus metaduomenis, kurie pateikti *.xml* failo formatu. Duomenų rinkinys suanotuotas rankiniu būdu su *labelImg* įrankiu sukuriant ribojantį langelį kiekvienam diagramos elementui eskize (3.3 pav.).



3.3 pav. Suanotuota diagrama *labelImg* įrankyje

Taigi, buvo sukurtas naujas PA diagramų duomenų rinkinys iš 346 eskizų, kuris buvo panaudotas realizuoti (2.3.1) diagramos simbolių atpažinimo komponentą. Tikimasi, kad duomenų rinkinį bus galima panaudoti įgyvendinant ir kitus tiriamuosius darbus susijusius su panaudojimo atvejų diagramomis.

3.3. Diagramos simbolių atpažinimo modelio apmokinimas

Diagramos simbolių atpažinimo modeliui realizuoti buvo naudojama *TensorFlow Object Detection API 2* programinė sąsaja. *TensorFlow* suteikia prieigą prie iš anksto apmokytų (angl. *pre – trained*) modelių su didelės apimties vaizdų duomenų rinkiniais, pavyzdžiui, *COCO* arba *ImageNet* [65]. Siekiant optimizuoti modelio veikimą, apmokinimo laiką buvo pasinaudota iš anksto apmokytais modeliais ir taikomi svorių, ir kitų modelio parametrų perkėlimo metodai (angl. *transfer learning*). Tam kaip bazinė architektūra buvo pasirinktas *Faster R-CNN ResNet152 V1 800x1333* neuroninis tinklas, kuris buvo naudojamas ir analizuotose sprendimuose (1.5). *Faster R-CNN* yra populiarus gilusis neuroninis tinklas naudojamas objektams aptikti nuotraukoje, ir kuris grąžina ribojančius langelius. Šis neuroninis tinklas yra populiarus dėl gana aukšto tikslumo ir greičio santykio, todėl kartais naudojamas net realaus laiko aplikacijose. *Faster R-CNN* modeliui buvo pritaikyta apmokymo konfigūracija pagal UML 2.5 panaudojimo atvejų diagramos simbolius ir kontekstą. Modelio apmokinimas buvo vykdomas ant GPU, o vieno apmokinimo trukmė iki 5 valandų. Buvo

eksperimentuojama su įvairiomis hyperparametrų reikšmėmis: partijos dydžiu (angl. *batch size*), apmokinimo žingsniu (angl. *learning step*), aktyvacijos funkcijomis, reguliarizacijos ir kitais parametrais, kol buvo rasta tokia hyperparametrų kombinacija su kuria pasiektas norimas modelio tikslumas. Apmokymo konfigūracijoje buvo sukonfigūruoti ir duomenų praplėtimo (angl. *data augmentation*) žingsniai, kurie padidina apmokinimo laiką, tačiau padeda modeliui pasiekti geresnį generalizacijos lygį. Duomenų praplėtimui naudotas atsitiktinis paveikslėlių horizontalus apsukimas ir atsitiktiktinis paveikslėlio apkarpymas. Apmokinimui duomenų rinkinio įvesties failai buvo konvertuojami į *TensorFlow 2.5* įvesties *.tfrecord* formatą. Duomenų rinkinys buvo skaidomas 90:10 santykiu apmokinimui ir testavimui.

3.4. Testavimas

UML panaudojimo atvejų diagramų eskizų skaitmenizavimo sistemai buvo vykdomas kiekvieno komponento rankinis testavimas. 3.4.1 poskyryje buvo testuojamas diagramos simbolių atpažinimo komponentas, 3.4.2 – ryšių regionų raktinių taškų nustatymo komponentas, 3.4.3 – OCR komponentas, 3.4.4 – diagramos konvertavimo į XMI failą komponentas 3.4.5 – naudotojo sąsaja.

3.4.1. Diagramos simbolių atpažinimo komponento testavimas

Diagramos simbolių atpažinimo komponento testavimas buvo vykdomas panaudojant testavimo imties duomenis modelio išvesties (angl. *inference*) žingsniui. Modeliui pateikus naują eskizą buvo tikrinamas gebėjimas prisitaikyti prie naujų diagramos eskizų, ribojančių langelių išsidėstymas, tikslumas. Testavimo metu buvo pastebėta, kad modelis neatpažįsta ortogonalių asociacijos, paveldėjimo ir priklausomybės ryšių. Dėl šios priežasties buvo praplėstas duomenų rinkinys, kuriame būtų padengtas tokių ryšių trūkumas. Apmokinus modelį dar kartą ortogonalūs ryšiai buvo atpažįstami teisingai. Toliau buvo testuojamas ribojančio langelio atitikimas diagramos simbolių elementams. Pastebėta, kad ryšių elementams ribojantis langelis yra didesnis negu ryšys, dėl ko gali būti mažesnis raktinių taškų nustatymo tikslumas regionų raktinių taškų nustatymo komponente. Ribojančio langelio neatitikimą lėmė nekorektiškai suanotuoti ryšių elementai duomenų rinkinyje, todėl rinkinys buvo peržiūrėtas ir pakoreguotas. Testavimo metu pastebėta, kad pasitaiko besikartojančių regionų tiems patiems diagramos elementams, arba nekorektiškai suklasifikuota regiono klasė. Dėl šios priežasties buvo realizuotas pasikartojančių elementų pašalinimas.

3.4.2. Ryšių regionų raktinių taškų nustatymo komponento testavimas

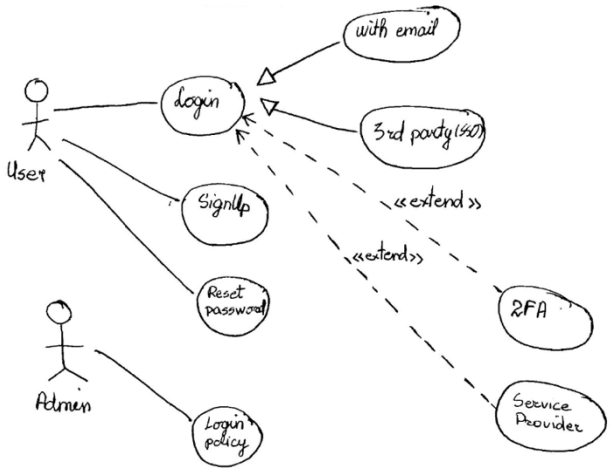
Ryšių regionų raktinių taškų nustatymo komponento testavimas buvo vykdomas pateikiant ryšio regioną komponentui ir tikrinant, kad pradžios ir pabaigos raktiniai taškai yra nustatyti teisingai. Pagrindiniai testavimo atvejai yra pateikti 3.4 pav. Raudonai pažymėtas taškas ribojančio langelio viduje yra pradžios taškas, o geltonai pažymėtas taškas – pabaigos. Visiems ryšio regionams tikrinama, kad raktiniai taškai yra teisinguose regiono kampuose ir, kad nustatyti du raktiniai taškai. Ortogonalių linijų regionams yra patikrinama, kad raktiniai taškai nustatomi per regiono trumpiausios kraštinės centrus abejuose galuose. Papildomai priklausomybės ir paveldėjimo ryšiui tikrinama, kad pabaigos tašku nustatomas tas ryšio regiono kampas, arba ta regiono kraštinė, kurioje yra rodyklės galvutė.

3.4 pav. Raktinių taškų nustatymo komponento testavimo atvejai

Kadangi raktiniai taškai nustatomi pagal priekinio plano pikselius testavimo metu pastebėta, kad neteisingai nustatomi raktiniai taškai, kai viename iš regiono kampų yra persidengiantis teksto regionas. Todėl buvo įvesta taisyklė, kad priekinio plano pikselių turi būti bent abejuose priešinguose regiono kampuose. Toliau testuojant raktinių taškų nustatymo komponentą buvo pastebėta, kad ne ortogonalioms linijoms kartais taikomas ortogonalių linijų raktinių taškų nustatymo algoritmas. Ši klaida buvo ištaisyta įvertinant ribojančio langelio matmenų proporciją su eskizu.

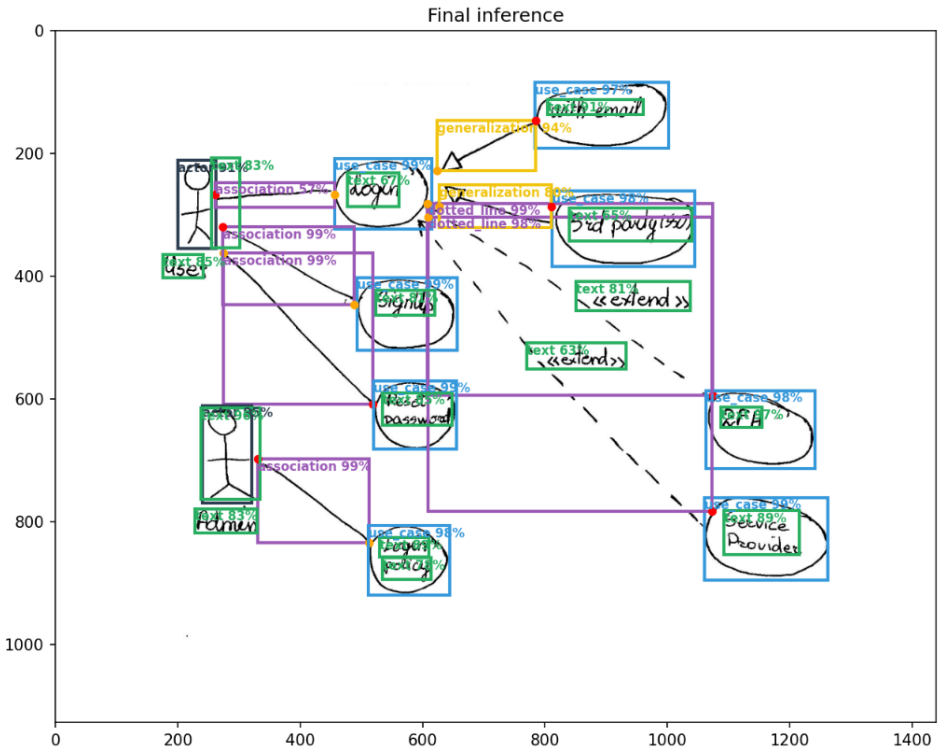
3.4.3. OCR komponento testavimas

OCR komponentas buvo testuojamas pateikiant diagramos eskizą komponentui ir tikrinant, kad grąžinti teisingi teksto regionai su skaitmenizuotu tekstu. 3.5 pav. yra pateiktas pavyzdinis PA diagramos eskizas, kuris pateikiamas OCR komponentui.



3.5 pav. PA diagramos eskizas OCR komponento testavimui

3.6 pav. yra pateiktas diagramos eskizas, kuriame pavaizduoti OCR komponento pateikti teksto regionai. Tikrinama, kad aktoriaus, panaudojimo atvejų ir priklausomybės ryšių pavadinimai yra pažymėti teksto regionais. Papildomai patikrinama, kad teksto regionų turinys atitiktų eskize matomus pavadinimus.

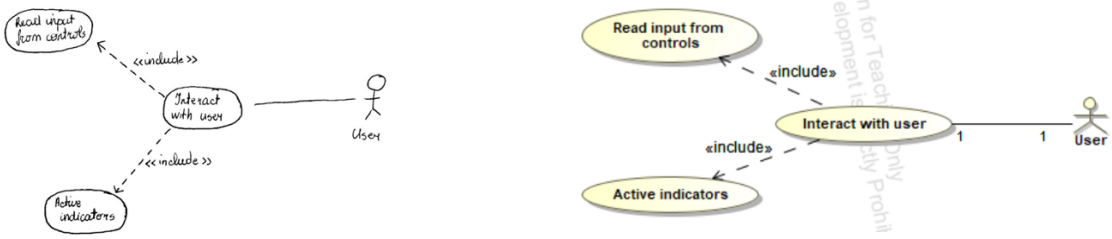


3.6 pav. OCR komponento grąžinti teksto regionai PA diagramos eskize

Testavimo metu pastebėta, kad „stick-man“ piktograma pažymėti aktoriaus elementai yra nustatomi kaip atskiri teksto regionai, todėl įvestas tokių regionų filtravimas.

3.4.4. Diagramos konvertavimo į XMI failą komponento testavimas

Diagramos konvertavimo į XMI failą komponento testavimas buvo vykdomas rankiniu būdu sugeneruotą XMI failą importuojant į *MSOSA* įrankį. Atidarius failą visi skaitmenizuoti diagramos elementai įkeliami į panaudojimo atvejo diagramos vizualizaciją ir pasirenkama rodyti visus ryšius tarp elementų. Tada diagramos elementai rankiniu būdu yra išdėliojami pagal pradinį eskizą ir tikrinami visi skaitmenizuoti elementai su pradiniu eskizu.

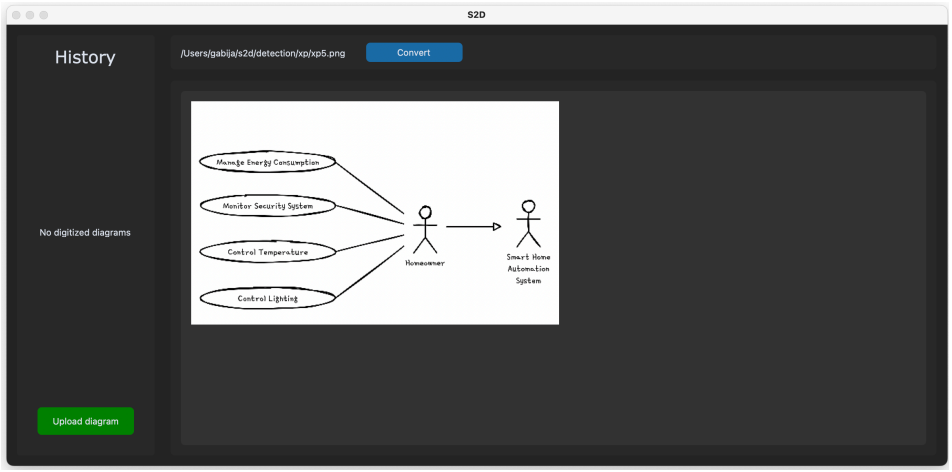


3.7 pav. Ranka braižytas ir skaitmenizuotas eskizas

3.7 pav. yra pateiktas ranka braižytas ir skaitmenizuotas eskizas. Patikrinama ar skaitmenizuoti visi panaudojimo atvejai, aktoriai ir ryšiai. Įsitikinama, kad priklausomybės ir paveldėjimo ryšiai yra sujungti tinkamomis kryptimis su teisingais elementais.

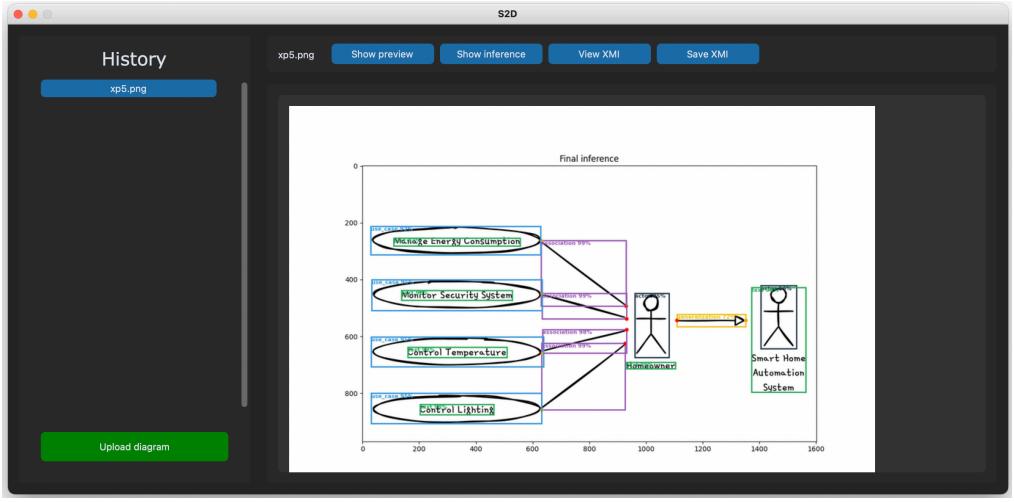
3.4.5. Naudotojo sąsajos testavimas

Naudotojo sąsajos testavimas buvo vykdomas rankiniu būdu. Ištestuoti pagrindiniai sistemos funkciniai reikalavimai: eskizo įkėlimas, skaitmenizavimo iniciavimas ir rezultato peržiūra, XMI peržiūra ir parsiuntimo iniciavimas. 3.8 pav. yra pateiktas pagrindinis sistemos langas su įkeltu diagramos eskizu. Diagramos eskizas yra įkeliamas paspaudus „*Upload diagram*“ mygtuką kairėje ekrano pusėje apačioje. Tada sistema pateikia eskizo įkėlimo langą ir pasirenkamas *.png* arba *.jpg* formato diagramos eskizo failas. Sėkmingai įkeltas diagramos eskizas pateikiamas dešinėje ekrano pusėje. Paspaudus „*Convert*“ mygtuką inicijuojamas eskizo skaitmenizavimas.



3.8 pav. Pagrindinis sistemos langas su įkeltu diagramos eskizu

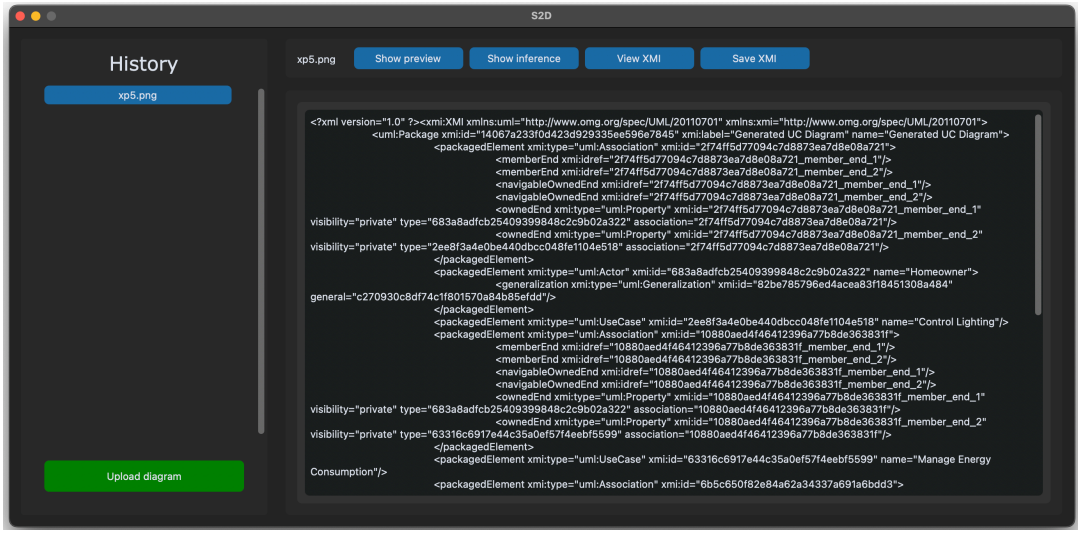
3.9 pav. yra pateiktas sistemos langas po sėkmingo eskizo skaitmenizavimo. Patikrinama, kad diagramos eskizas yra įtraukiamas į istorijos sąrašą kairėje ekrano pusėje. Toliau paspaudus mygtuką „*Show inference*“ peržiūrimas sėkmingai skaitmenizuotos diagramos rezultatas su pažymėtais regionais.



3.9 pav. Sistemos langas po sėkmingo eskizo skaitmenizavimo

3.10 pav. yra pateiktas XMI failo turinio peržiūros langas, kuris atidaromas paspaudus mygtuką „*View XMI*“. Patikrinama, kad paspaudus mygtuką „*Save XMI*“ inicijuojamas XMI failo

parsiuntimas. Toliau skaitmenizuojamas antras diagramos eskizas. Patikrinama, kad istorijos sąraše yra įtrauktos abi skaitmenizuotos diagramos ir galima peržiūrėti senesnius įrašus.



3.10 pav. XMI failo turinio peržiūros langas

Taigi 3.4 skyrelyje aprašyta, kaip buvo vykdomas rankinis kiekvieno komponento testavimas. Kitame skyriuje (4) aprašomas tiriamojo darbo eksperimentas ir rezultatai.

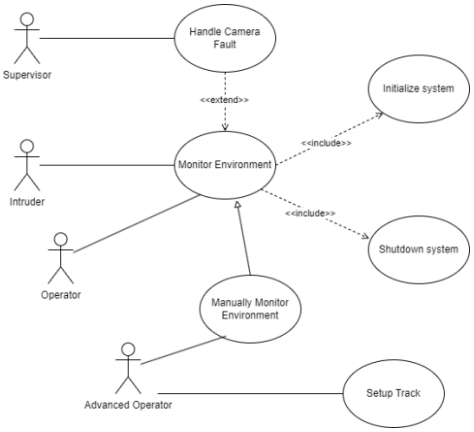
4. Eksperimentinis UML diagramų eskizų skaitmeninimo tyrimas

4.1. Eksperimento planas

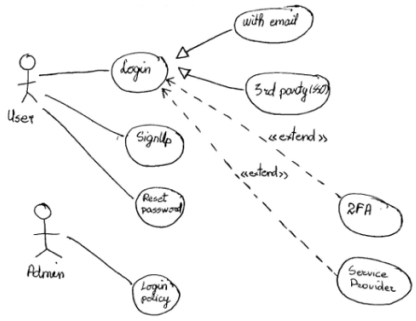
4.2 skyrelyje aprašytas eksperimentui vykdyti sudarytas naujas duomenų rinkinys. 4.3 skyrelyje aprašyta eksperimento vykdymo eiga ir pagrindinės naudojamos metrikos įvertinti rezultatus. UML diagramų eskizų skaitmeninimo eksperimentas yra sudarytas iš dviejų dalių. Pirmoje dalyje yra vykdomas diagramos simbolių atpažinimo eksperimentas. Antrojoje dalyje yra vykdomas diagramos skaitmenizavimo eksperimentas, kurio metu įvertinama kaip tiksliai skaitmenizuojami ir sujungiami elementai į vientisą diagramos struktūrą.

4.2. Eksperimento duomenų rinkinys

Diagramos simbolių atpažinimo tikslumo įvertinimo eksperimentui buvo sudarytas naujas PA diagramų rinkinys, kurį sudaro 32 eskizai. 17 (53.12%) eskizų nubraižyti su įvairias įrankiais, o 15 (46.88%) eskizų nubraižyti ranka ant popieriaus. Diagramų rinkinį eksperimentui sudarė 3 asmenys. 4.1 pav. ir 4.2 pav. yra pavaizduotos diagramos nubraižytos įrankiu ir ranka. Įrankiu braižytos diagramos sukurtos įrankiais *draw.io* ir *Visual Paradigm*. Dalis diagramų buvo rasta internete. Ranka braižytos diagramos buvo braižomos ant popieriaus ir skenuojamos. Pagrindiniai skirtumai tarp įrankiu ir ranka braižytų diagramų yra tokie, kad ranka braižytuose diagramose ryšiai nėra visiškai tiesūs, aktorių ir panaudojimo atvejų elementai kažkiek skiriasi kiekvienoje diagramoje ir nėra nubraižyti vienodai, elementų pavadinimai yra rašytinėmis raidėmis.



4.1 pav. Įrankiu nubraižytos PA diagramos pavyzdys



4.2 pav. Ranka nubraižytos PA diagramos pavyzdys

Iš viso diagramų rinkinyje yra 763 elementai, kurių pasiskirstymo statistika yra pateikiama 4.1 lentelė. Teksto elementų duomenų rinkinyje yra 299 (38.9%), asociacijos ryšio elementų – 186 (24.5%), panaudojimo atvejų elementų – 166 (21.7%), aktoriaus elementų – 75 (9.8%), paveldėjimo ryšio elementų – 39 (5.1%). Vykdant abu eksperimentus buvo naudojamos tokios klasės patikimumų reikšmės: teksto klasei – 60%, asociacijos ryšiui – 20%, panaudojimo atvejui – 30%, aktoriui – 35%, paveldėjimo ryšiui – 20%. Suklasifikuoti elementai, kurių patikimumas mažesnis negu priimtinas, yra atmetami.

4.1 lentelė. Eksperimento duomenų rinkinys

Klasė	**Kiekis**	**Procentinė išraiška**	**Naudotas klasės patikimumas**

Tekstas	299	38,9%	60%
Asociacijos ryšys	186	24.5%	20%
Panaudojimo atvejis	166	21.7%	30%
Aktorius	75	9.8%	35%
Paveldėjimo ryšys	39	5.1%	20%

Diagramų rinkinį sudaro tik korektiškos diagramos pagal *UML 2.5* specifikaciją. Stengtasi sudaryti tokias diagramas, kurios galėtų būti nubraižomos realiomis sąlygomis.

4.3. Bendra vykdymo eiga ir naudojamos metrikos

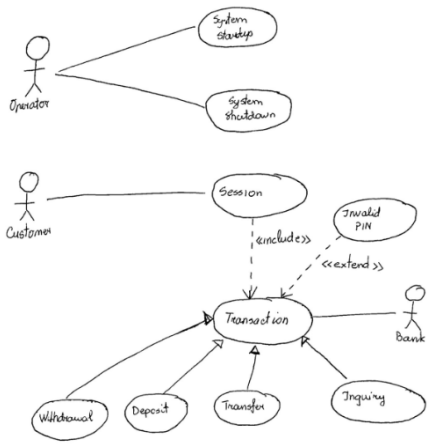
Eksperimentas buvo vykdomas rankiniu būdu, skaitmenizuojant kiekvieną diagramos eskizą su sukurtu įrankiu. Kiekvienai diagramai buvo sudaroma lentelė, kurioje paskaičiuojami kiekvieno elemento tikrasis teigiamas rezultatas TP (angl. *True positive*), klaidingas neigiamas rezultatas FN (angl. *False negative*), klaidingai teigiamas rezultatas FP (angl. *False positive*). TP reikšmė reiškia, kad diagramos elementas buvo teisingai atpažintas, FN – diagramos elementas buvo praleistas (neatpažintas), FP – atpažintas neteisingas (perteklinis) diagramos elementas, kurio nėra diagramos eskize. Iš apskaičiuotų TP, FN ir FP reikšmių skaičiuojamos atkūrimo (angl. *recall*) ir preciziškumo (angl. *precision*) reikšmės. Atkūrimo reikšmė yra skaičiuojama pagal formulę $Recall = \frac{TP_{Class\ A}}{TP_{Class\ A} + FN_{Class\ A}}$ ir reiškia gebėjimą nustatyti visus tam tikros klasės atvejus. Preciziškumo reikšmė yra skaičiuojama pagal formulę $Precision = \frac{TP_{Class\ A}}{TP_{Class\ A} + FP_{Class\ A}}$ ir reiškia gebėjimą teisingai nustatyti tam tikros klasės atvejus. Preciziškumo ir atkūrimo metrikos padeda įvertinti skirtingus našumo aspektus: preciziškumu vertinamas teigiamų prognozių teisingumas, o atkūrimu - gebėjimas užfiksuoti visus teigiamus atvejus. Apskaičiavus atskirų klasių atkūrimo ir preciziškumo metrikas įvertinamas bendras diagramos atkūrimas ir preciziškumas. Tam naudojama makrovidurkio (angl. *Macro – averaging*) metrika. Preciziškumo ir atkūrimo makrovidurkiai apskaičiuojami sudėjus atskirų klasių preciziškumo ir atkūrimo reikšmes ir padalinus iš klasių skaičiaus: $Precision_{Macro-average} = \frac{Precision_{class\ A} + Precision_{class\ B} + \cdots + Precision_{class\ N}}{N}$, $Recall_{Macro-average} = \frac{Recall_{class\ A} + Recall_{class\ B} + \cdots + Recall_{class\ N}}{N}$. Galimos reikšmės yra nuo 0 iki 1. Makrovidurkio metrika buvo pasirinkta todėl, kad visos klasės PA diagramų skaitmenizavimo uždavinyje yra vienodai svarbios, ir todėl, kad turimas nesubalansuotas duomenų rinkinys. Turint atkūrimo ir preciziškumo makrovidurkų įvertinimus papildomai buvo apskaičiuotas F1 statistikos (angl. *F1 – score*) įvertis. Šis įvertis yra harmoningas preciziškumo ir atkūrimo vidurkis, kuris naudojamas klasifikavimo modelio tikslumui vertinti, atsižvelgiant tiek į klaidingai teigiamus, tiek į klaidingai neigiamus rezultatus. F1 statistikos įvertis skaičiuojamas pagal formulę: $F1_{Score} = \frac{2 * Precision * Recall}{Precision + Recall}$.

4.4. Diagramos simbolių atpažinimo eksperimentas

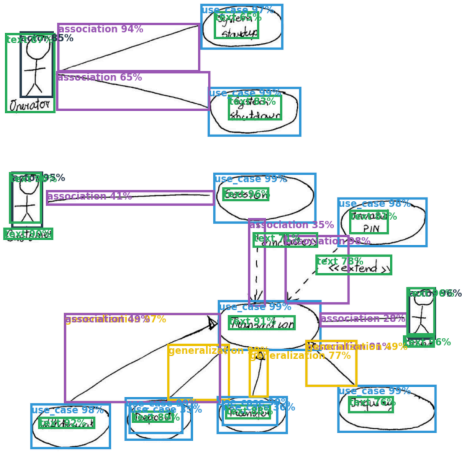
4.4.1. Vykdymo eiga

4.3 pav. yra pavaizduotas ranka braižyto diagramos eskizo egzempliorius, kuris buvo naudojamas eksperimento metu. Pateiktame diagramos eskize yra 3 aktoriai, 9 panaudojimo atvejai, 4 paprastos asociacijos ryšiai, 1 apėmimo ryšys, 1 išplėtimo ryšys, 4 paveldėjimo ryšiai ir 14 teksto elementų. Toliau diagramos eskizas yra skaitmenizuojamas ir suskaičiuojami atpažinti diagramos simboliai. 4.4

pav. yra pateiktas skaitmenizuotos diagramos eskizo egzempliorius. Peržiūrėjus skaitmenizavimo rezultatą apskaičiuojamas atkūrimo ir preciziškumo įvertinimas kiekvienai diagramos elemento klasei ir bendras elementų atpažinimo atkūrimas ir preciziškumas.



4.3 pav. Diagramos eskizo egzempliorius panaudotas eksperimente



4.4 pav. Skaitmenizuotas diagramos eskizo egzempliorius

4.2 lentelė yra pateikta skaitmenizuoto diagramos eskizo klasių ir bendras diagramos įvertinimas. Iš viso buvo rasti 6 asociacijos ryšiai (TP), 0 praleistų ryšių (FN) ir 1 papildomas asociacijos ryšys (FP). Asociacijos ryšio atkūrimas yra vertinamas 1, o preciziškumas 0.86. Svarbu paminėti, kad apėmimo ryšys įeina į asociacijos ryšio klasę atpažinimo etape (žr. Priklausomybės ryšio nustatymas). Toliau tikrinamas paveldėjimo ryšio atpažinimo preciziškumas. Rasti 4 teisingi paveldėjimo ryšiai (TP) kaip ir pradiniame eskize ir nebuvo praleistų (FN) ar papildomai nustatytų paveldėjimo ryšio elementų (FP). Tokiu atveju šios klasės atkūrimas ir preciziškumas vertinami 1. Teisingi (TP) aktoriaus elementai nustatyti 3 be praleistų (FN) ir papildomų (FP) aktoriaus elementų, todėl atkūrimas ir preciziškumas vertinami 1. Kaip matoma iš 4.4 pav. rasti visi (TP) panaudojimo atvejo elementai ir 2 papildomi (FP), todėl atkūrimas yra lygus 1, o preciziškumas – 0.82. Teksto elementų teisingai nustatyta (TP) 14 iš 14 ir 2 papildomi (FP) elementai, todėl atkūrimas lygus 1, o preciziškumas – 0.88. Galiausiai apskaičiuojami atkūrimo ir preciziškumo makrovidurkiai. Visų elementų atkūrimo makrovidurkis yra 1, o preciziškumas 0.91.

4.2 lentelė. Skaitmenizuoto diagramos eskizo klasių atpažinimo atkūrimas ir preciziškumas

Klasė/Metrika	**Elementai**	**TP**	**FN**	**FP**	**Atkūrimas**	**Preciziškumas**
Asociacijos ryšys	6	6	0	1	1,00	0,86
Paveldėjimo ryšys	4	4	0	0	1,00	1,00
Aktorius	3	3	0	0	1,00	1,00
Panaudojimo atvejis	9	9	0	2	1,00	0,82
Tekstas*	14	14	0	2	1,00	0,88
Bendras tikslumas	**Makrovidurkiai**					
	Atkūrimas	**Preciziškumas**				

Visi elementai	1	0.91

Taigi, kiekvienai diagramai iš sudaryto rinkinio buvo apskaičiuotos atkūrimo ir preciziškumo metrikos bei šių metrikų makrovidurkiai kaip pavaizduota 4.2 lentelė. Kitame skyrelyje (4.4.2) aptariami diagramos simbolių atpažinimo eksperimento rezultatai.

4.4.2. Rezultatai

Apskaičiavus kiekvienos diagramos rezultatus buvo apskaičiuotas viso diagramos rinkinio atkūrimo ir preciziškumo metrikos per klasę, ir šių metrikų makrovidurkiai siekiant įvertinti bendrą simbolių atpažinimo atkūrimą ir preciziškumą. 4.3 lentelė yra pateikti diagramos simbolių atpažinimo eksperimento rezultatai. Asociacijos ryšio (įeina apėmimo ir išplėtimo priklausomybės ryšiai) elemento atkūrimas yra 0.95, o preciziškumas 0.85. Teisingai buvo atpažinti 177 elementai iš 186 ir 9 asociacijos ryšio elementai neatpažinti. Papildomai buvo neteisingai nustatyta 32 asociacijos ryšių, kas lemia tokį šio elemento preciziškumą. Paveldėjimo ryšio atkūrimas yra 0.90, o preciziškumas 0.90. Teisingai buvo atpažinti 35 paveldėjimo ryšiai iš 39, 4 paveldėjimo ryšiai nebuvo nustatyti ir rasti 4 papildomi ryšiai. Aktoriaus preciziškumas ir atkūrimas yra 1. Visi 75 aktoriaus elementai buvo nustatyti teisingai ir nebuvo rasta nei vieno papildomo elemento. Panaudojimo atvejo elemento atkūrimas yra 1, o preciziškumas 0.95. Teisingai nustatyti visi 166 panaudojimo atvejų elementai ir papildomai neteisingai nustatyti 8 elementai. Mažiausiu preciziškumu pasižymi teksto klasės elementai – 0.85, tačiau atkūrimas kaip ir kitų klasių yra aukštas – 0.97. Teisingai buvo nustatyti 289 teksto elementai iš 299, 10 teksto elementai buvo nenustatyti ir 51 nustatyti papildomai.

4.3 lentelė. Diagramos simbolių atpažinimo tikslumo eksperimento rezultatai

Klasė/Metrika	**Elementai**	**TP**	**FN**	**FP**	**Atkūrimas**	**Preciziškumas**
Asociacijos ryšys	186	177	9	32	0.95	0.85
Paveldėjimo ryšys	39	35	4	4	0.90	0.90
Aktorius	75	75	0	0	1,00	1,00
Panaudojimo atvejis	166	166	0	8	1,00	0,95
Tekstas*	299	289	10	51	0,97	0,85
Bendras tikslumas	**Makrovidurkiai**		**F1 statistika**			
	Atkūrimas	**Preciziškumas**	0.94			
Visi elementai	0,96	0.91				

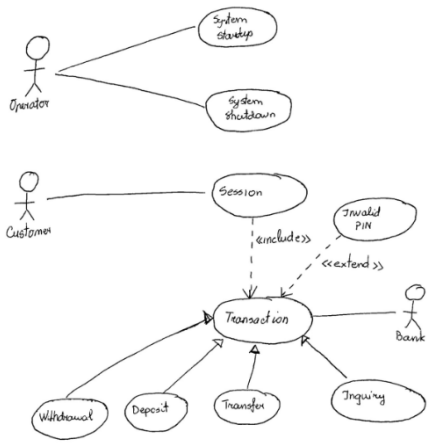
Atkūrimas visoms klasėms yra gana aukštas ir svyruoja nuo 0.90 paveldėjimo ryšiui iki 1 aktoriaus ir panaudojimo atvejų elementams. Asociacijos ryšio ir teksto elementai atitinkamai pasižymi 0.95 ir 0.97 atkūrimu. Kita vertus, preciziškumas buvo gautas mažesnis negu atkūrimas ir svyruoja nuo 0.85 teksto ir asociacijos elementams iki 1 aktoriui. Asociacijos ryšio, paveldėjimo ryšio ir panaudojimo atvejų elementų preciziškumas atitinkamai yra 0.85, 0.90 ir 0.95. **Taigi, bendras visų elementų atkūrimas yra 0.96, o preciziškumas – 0.91. F1 statistikos įvertinimas yra 0.94.** Verta paminėti, kad preciziškumo ir atkūrimo metrikos padeda įvertinti skirtingus našumo aspektus: preciziškumu vertinamas teigiamų prognozių teisingumas, o atkūrimu - gebėjimas užfiksuoti visus teigiamus

atvejus. Neretai didinant atkūrimą sumažėja preciziškumas ir atvirkščiai. Žemesnis preciziškumas gautas todėl, kad elementai buvo priimami su ganėtinai mažu patikimumu (4.1 lentelė), pavyzdžiui, asociacijos ir paveldėjimo ryšiai priimami tik su 20% patikimumu, kas neišvengiamai reiškia, jog bus atpažinti ne tik elementai, kurie iš tikrųjų egzistuoja eskize (TP), bet ir neteisingai nustatyti elementai, kurių eskize nėra (FP). Visgi, šiame uždavinyje svarbiau gauti aukštesnį atkūrimo įvertinimą negu preciziškumą. Kitaip tariant, svarbiau, kad būtų nustatyta kuo daugiau egzistuojančių elementų, negu jie būtų praleisti dėl pasirinkto aukštesnio patikimumo.

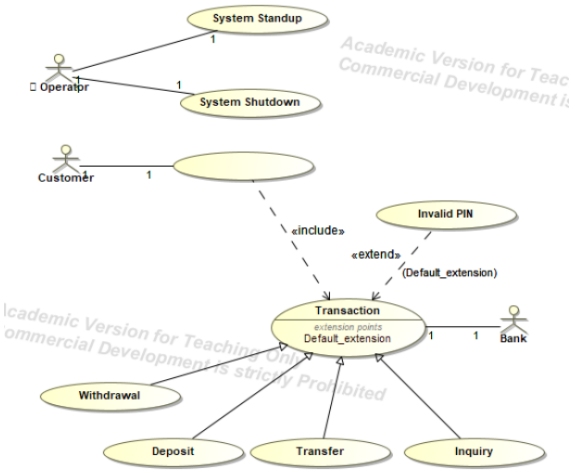
4.5. Diagramos skaitmenizavimo eksperimentas

4.5.1. Vykdymo eiga

Diagramos skaitmenizavimas, be elementų atpažinimo, susideda iš daug papildomų žingsnių: ryšių raktinių taškų nustatymo, elementų pavadinimų nustatymo, pasikartojančių elementų pašalinimo, diagramos elementų sujungimo, priklausomybės ryšio patikslinimo, serializavimo į XMI formatą ir kt. Todėl, norint įvertinti diagramos skaitmenizavimo atkūrimą ir preciziškumą atliekamas rankinis įvertinimas lyginant įrankio skaitmenizuotą diagramą su pradiniu diagramos eskizu. 4.5 pav. yra pavaizduotas neskaitmenizuotas diagramos eskizo egzempliorius, kuris buvo naudotas eksperimento metu.



4.5 pav. Neskaitmenizuotas diagramos eskizas



4.6 pav. Skaitmenizuotas diagramos eskizas *MSOSA* įrankyje

Pradiniame diagramos eskize yra 4 aktoriai, 6 panaudojimo atvejai, 5 asociacijos ryšiai, 1 paveldėjimo ryšys, 2 apėmimo ryšiai ir 1 išplėtimo ryšys. Diagrama yra skaitmenizuojama ir gautas XMI failas importuojamas į *MSOSA* įrankį. Atidarius failą visi skaitmenizuoti diagramos elementai įkeliami į panaudojimo atvejų diagramos vizualizaciją ir pasirenkama rodyti visus ryšius tarp elementų. Tada diagramos elementai rankiniu būdu yra išdėliojami pagal pradinį eskizą ir tikrinami visi skaitmenizuoti elementai su pradiniu eskizu. 4.6 pav. yra pateiktas skaitmenizuotos diagramos eskizas importuotas *MSOSA* įrankyje. Diagramos skaitmenizavimo rezultato įvertinimas susideda iš teisingai skaitmenizuotų diagramos elementų ir teisingai sujungtų elementų atšaukimo ir tikslumo metrikų. 4.4 lentelė yra pateikti diagramos elementų skaitmenizavimo rezultatai. Visų elementų, išskyrus teksto, atkūrimas ir preciziškumas yra 1. Kaip matoma 4.6 pav. vieno panaudojimo atvejo elemento pavadinimas nebuvo teisingai skaitmenizuotas, dėl to atkūrimas yra lygus 0.93. Bendras šios diagramos elementų skaitmenizavimo atkūrimas yra 0.99, o preciziškumas 1. Galima pastebėti,

kad preciziškumas skaitmenizavimo etape pagerėjo nuo 0.91 (4.4.1) iki 1. Preciziškumo metrika yra pagerinama todėl, kad atliekamas papildomas atpažintų diagramos simbolių apdorojimas (žr. ***Papildomo apdorojimo žingsniai tikslumui pagerinti***). Svarbu paminėti, kad skaičiuojant teksto elementų skaitmenizavimo preciziškumą buvo leistina 1 – 4 simbolių paklaida. Toks sprendimas priimtas todėl, kad dėl keletos papildomų arba praleistų simbolių vistiek galima suprasti elemento pavadinimą, arba greitai atlikti pataisymą rankiniu būdu. Beto, tam tikrus ranka rašytus pavadinimus gali būti sunku teisingai atpažinti ir žmogui. Nustatytas aktoriaus pavadinimas „[]Operator“ vietoj „Operator“ ar nustatytas panaudojimo atvejo pavadinimas „System Standup“ vietoj „System Startup“ nėra laikomi klaidomis skaičiuojant atšaukimo ir tikslumo įverčius.

4.4 lentelė. Diagramos elementų skaitmenizavimo rezultatai.

Klasė/Metrika	**Elementai**	**TP**	**FN**	**FP**	**Atkūrimas**	**Preciziškumas**
Asociacijos ryšys	4	4	0	0	1,00	1,00
Išplėtimo ryšys	1	1	0	0	1,00	1,00
Apėmimo ryšys	1	1	0	0	1,00	1,00
Paveldėjimo ryšys	4	4	0	0	1,00	1,00
Aktorius	3	3	0	0	1,00	1,00
Panaudojimo atvejis	9	9	0	0	1,00	1,00
Tekstas	14	13	1	0	0,93	1,00

Bendras tikslumas	**Makrovidurkiai**	
	Atkūrimas	**Preciziškumas**
Visi elementai	0,99	1,00

4.5 lentelė. yra pateikti diagramos skaitmenizuotų elementų sujungimo rezultatai. Teisingai (TP) skaitmenizuotas ryšys yra laikomas toks, kai ryšio galuose yra teisingi elementai ir kryptis yra teisinga. Neteisingai (praleistu) ryšiu yra laikomas toks ryšys, kuris arba turi susietus neteisingus elementus ryšio gale, arba ryšio galuose yra teisingi elementai, bet neteisinga kryptis. Papildomai nustatytu (FP) ryšiu yra laikomas toks ryšys, kurio nebuvo pradiniame diagramos eskize. Šioje skaitmenizuotoje diagramoje visi asociacijos, išplėtimo, apėmimo ir paveldėjimo ryšiai yra sujungti teisingai. Kryptys irgi yra teisingos, todėl atkūrimo ir preciziškumo metrikos šiems ryšiams atitinkamai siekia 1. Bendras visų sujungimų atkūrimas ir preciziškumas yra 1.

4.5 lentelė. Diagramos skaitmenizuotų elementų sujungimo rezultatai.

Ryšių sujungimai/Metrika	**Sujungimai**	**TP**	**FN**	**FP**	**Atšaukimas**	**Tikslumas**
Asociacijos ryšys	4	4	0	0	1,00	1,00
Išplėtimo ryšys	1	1	0	0	1,00	1,00
Apėmimo ryšys	1	1	0	0	1,00	1,00
Paveldėjimo ryšys	4	4	0	0	1,00	1,00

Bendras tikslumas	**Makrovidurkiai**	
	Atšaukimas	**Tikslumas**

Visi sujungimai	1,00	1,00

Kiekvienai eksperimento rinkinio diagramai buvo apskaičiuoti diagramos elementų skaitmenizavimo rezultatai ir diagramos skaitmenizuotų elementų sujungimo rezultatai. Kitame skyrelyje (0) aptariami diagramos skaitmenizavimo eksperimento rezultatai.

4.6. Rezultatai

Apskaičiavus kiekvienos diagramos elementų skaitmenizavimo ir elementų sujungimo rezultatus buvo apskaičiuotos viso diagramos rinkinio atkūrimo ir preciziškumo metrikos per klasę, ir šių metrikų makrovidurkiai. 4.6 lentelė yra pateikti diagramos skaitmenizavimo eksperimento elementų skaitmenizavimo rezultatai. Iš viso teisingai skaitmenizuota 125 asociacijos ryšio elementų iš 134, 9 ryšiai buvo neskaitmenizuoti ir 9 ryšiais buvo skaitmenizuota per daug. Atkūrimo įvertinimas yra 0.93, o preciziškumas – 0.93. Išplėtimo ryšio atkūrimas yra 0.85, o preciziškumas 0.92. Teisingai skaitmenizuoti 22 išplėtimo ryšiai iš 26 ir 4 išplėtimo ryšiai buvo praleisti. Paveldėjimo ryšių teisingai skaitmenizuota 33 iš 39 ir 6 ryšiai buvo praleisti, kas lemia 0.85 atkūrimą ir preciziškumą 0.97. Visi apėmimo ryšiai ir aktoriai skaitmenizuoti teisingai, todėl šių elementų atkūrimas ir preciziškumas yra 1. Panaudojimo atvejo elemento skaitmenizavimo rezultatai taip pat aukšti. Teisingai skaitmenizuoti visi 166 iš 166 panaudojimo atvejų elementai ir tik vienas elementas skaitmenizuotas papildomai. Panaudojimo atvejo elemento skaitmenizavimo atkūrimas yra 1, o preciziškumas – 0.99. Teksto elementų skaitmenizavimo atkūrimas yra 0.90, o preciziškumas – 0.99. Mažesnį teksto skaitmenizavimo atkūrimo įvertinimą lemia tai, kaip nustatomi teksto elementai diagramos simbolių komponente. Pavyzdžiui, panaudojimo atvejo elemento ribojančio langelio viduje, gali būti nustatyti 2 teksto blokai, tačiau skaitmenizuojant jie neapjungiami į vieną elemento pavadinimą, todėl gali būti neskaitmenizuojamas 1-2 pavadinimo žodžiai, o tai laikoma klaida. Taip pat, galima pastebėti, kad palyginus su pirmuoju eksperimentu beveik visų elementų skaitmenizavimo preciziškumas išaugo nuo mažiausio 0.85 iki 0.92. Tokį preciziškumo įverčio padidėjimą galima paaiškinti realizuotu papildomu apdorojimu po diagramos elementų atpažinimo žingsnio. **Taigi, bendras visų elementų skaitmenizavimo atkūrimas yra 0.93, o preciziškumas 0.97. F1 statistikos įvertinimas yra 0.95.**

4.6 lentelė. Diagramos skaitmenizavimo eksperimento elementų skaitmenizavimo rezultatai

Klasė/Metrika	**Elementai**	**TP**	**FN**	**FP**	**Atkūrimas**	**Preciziškumas**
Asociacijos ryšys	134	125	9	9	0,93	0,93
Išplėtimo ryšys	26	22	4	2	0,85	0,92
Apėmimo ryšys	26	26	0	0	1,00	1,00
Paveldėjimo ryšys	39	33	6	1	0,85	0,97
Aktorius	75	75	0	0	1,00	1,00
Panaudojimo atvejis	166	166	0	1	1,00	0,99
Tekstas	299	267	32	2	0,90	0,99
Bendras tikslumas	**Makrovidurkiai**		**F1 statistika**			
	Atkūrimas	**Preciziškumas**	0,95			
Visi elementai	0,93	0,97				

4.7 lentelė yra pateikti diagramos skaitmenizavimo eksperimento elementų sujungimo rezultatai. Diagramoje iš viso buvo panaudota 134 asociacijos ryšių sujungimų iš kurių 125 nustatyti teisingai ir 9 nustatyti neteisingai, nes ryšiai nebuvo skaitmenizuoti. Elementai buvo neteisingai sujungti 8 asociacijos ryšiais. Asociacijos ryšio atkūrimas ir preciziškumas yra 0.93, o preciziškumas – 0.94. Išplėtimo ryšio elementai buvo skaitmenizuota 16 iš 26, 10 iš jų buvo sujungti ne su tais panaudojimo atvejais, arba ne ta kryptimi. Šio ryšio atkūrimas yra mažiausias 0.62, o preciziškumas – 0.84. Apėmimo ryšiu teisingai sujungti visi elementai, todėl atkūrimas ir preciziškumas atitinkamai yra 1. Paveldėjimo ryšiu buvo teisingai sujungti 32 elementai, 6 elementai turėjo neteisingą kryptį ir 1 paveldėjimo ryšys sujungė ne tuos elementus. Paveldėjimo ryšio atšaukimas yra 0.82, o tikslumas – 1. **Taigi, bendras visų ryšių sujungimo atšaukimo įvertinimas yra 0.84, o tikslumas 0.95. F1 statistikos įvertinimas yra 0.89.** Buvo pastebėta, kad diagramos simbolių atpažinimo modelis teksto bloką *„<<extend>>“* kartais nustato, kaip asociacijos ryšį, būtent dėl to šio ryšio tipo atkūrimas yra mažiausias. Paveldėjimo ryšio atkūrimas yra 0.82, dėl to kad neteisingai nustatomi pradžios ir pabaigos raktiniai taškai dėl persidengiančio aktoriaus pavadinimo ant ryšio galvutės. Taip pat, pastebėta, kad diagramos simbolių atpažinimo etape tiesūs (ortogonalūs 2.19 pav.) ryšiai yra dažniausiai atpažįstami su mažesniu patikimumu, o kai kurie iš šių ryšių nesiekia net 20% patikimumo, kas lemia ryšio elementų mažesnį atkūrimą ir preciziškumą.

4.7 lentelė. Diagramos skaitmenizavimo eksperimento elementų sujungimo rezultatai

Ryšių sujungimai/Metrika	**Sujungimai**	**TP**	**FN**	**FP**	**Atkūrimas**	**Preciziškumas**
Asociacijos ryšys	134	125	9	8	0,93	0,94
Išplėtimo ryšys	26	16	10	3	0,62	0.84
Apėmimo ryšys	26	26	0	0	1,00	1,00
Paveldėjimo ryšys	39	32	7	0	0,82	1,00
Bendras tikslumas	**Makrovidurkiai**		**F1 statistika**			
	Atkūrimas	**Preciziškumas**	0,89			
Visi sujungimai	0,84	0,95				

Apibendrinus, atskiri elementai yra skaitmenizuojami gana aukštu atkūrimu (0.93) ir preciziškumu (0.97). Tokia atkūrimo metrika reiškia, kad 93% elementų buvo skaitmenizuoti teisingai, o 97% preciziškumas reiškia, kad buvo skaitmenizuoti tik tie 93% elementų, kurie egzistavo diagramos eskize. 3% visų skaitmenizuotų elementų buvo nustatyta papildomai, kurie neegzistavo eskize.

Gauti rezultatai sutampa su analizuotais egzistuojančiais sprendimais (1.11 lentelė). *DrawNet* ir *Arrow R-CNN* sprendimų diagramos simbolių atpažinimo tikslumas siekia 98.4% - 99.5%. Šio sprendimo atkūrimą būtų galima pagerinti sudarius didesnį duomenų rinkinį ir padidinus patikimumo ribas. Kita vertus, diagramos simbolių sujungimo atkūrimas siekia tik 0.84 su 0.95 preciziškumu. Tai reiškia, kad 16% diagramos elementų buvo nesujungti, arba sujungti neteisingais elementais, arba sujungti neteisinga kryptimi. Sujungimo atkūrimą ir preciziškumą daugiausiai lemia ryšių raktinių taškų nustatymo komponentas, kurį patobulinus būtų gautas geresnis elementų sujungimo rezultatas.

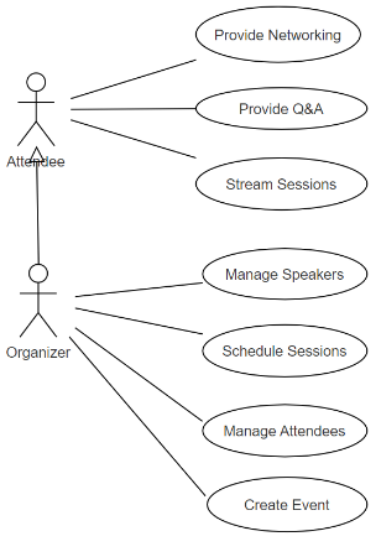
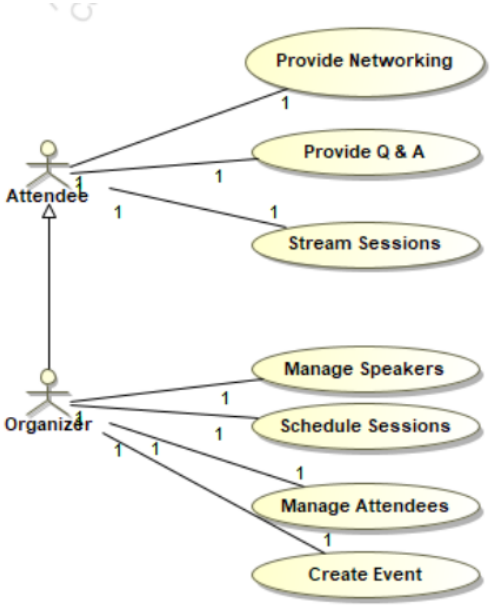
4.7. Sprendimo taikymo rekomendacijos

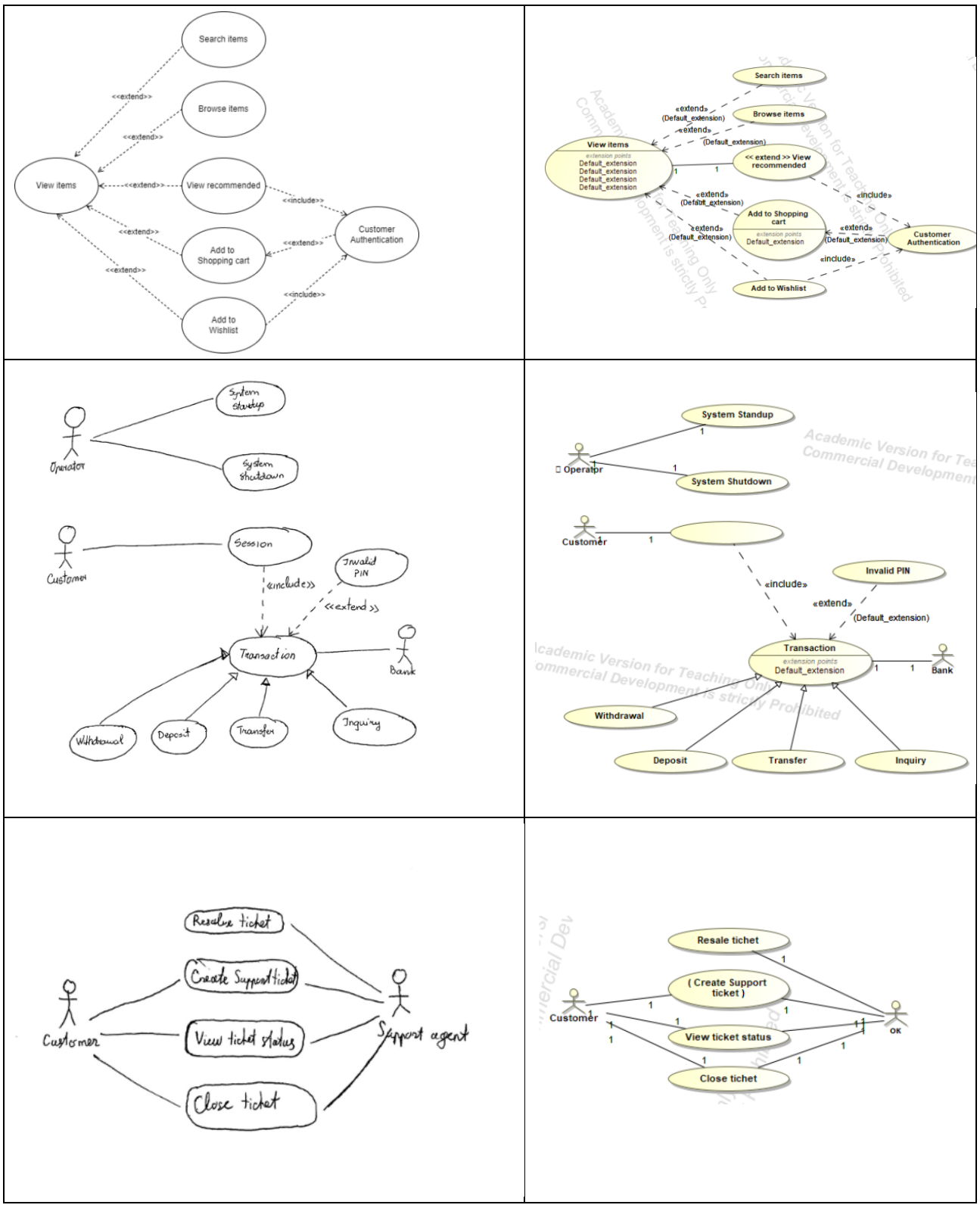
4.7.1. Sprendimo pritaikymas

Įvairiose dalykinėse srityse UML panaudojimo atvejų eskizai vis dar gali būti braižomi be CASE diagramas palaikančio įrankio - ranka ant popieriaus, išmaniosios lentos arba su laisvos formos braižymo įrankiais kaip *draw.io*. Siekiant toliau plėtoti pradinį eskizą ir patogiai pasidalinti su kitais suinteresuotais asmenimis, diagramos eskizas turi būti perkeltas į UML standartą palaikantį įrankį rankiniu būdu. Toks procesas yra neefektyvus, reikalaujantis daug laiko, atsiranda galimybė padaryti klaidų. UML panaudojimo atvejų eskizų skaitmenizavimo įrankis, sukurtas tiriamojo darbo metu, gali būti naudojamas siekiant palengvinti sistemų projektavimą automatizuojant UML panaudojimo atvejų diagramų eskizų skaitmenizavimą. 4.8 lentelė yra pateikta ranka ir su įrankiu braižytų diagramos eskizo pavyzdžių, kurie buvo sėkmingai skaitmenizuoti. Visi bandymai buvo atlikti su *MSOSA* sistema. Su įrankiu galima sėkmingai skaitmenizuoti diagramas, kurios yra:

- neutraliame baltame fone, kur nėra šešėlių ar pašalinių elementų nesusijusių su PA diagrama;
- be persidengiančių ir netiesių ryšių;
- sudarytos pagal *UML 2.5* specifikaciją;
- su įskaitomais elementų pavadinimais;
- aukštos rezoliucijos.

4.8 lentelė. Įrankio sėkmingai skaitmenizuojamų diagramų pavyzdžiai

Pradinis diagramos eskizas	**Skaitmenizuotas eskizas ir įkeltas į *MSOSA* įrankį**
	



Skaitmenizavus diagramą ir atidarius XMI failą užkraunami skaitmenizuoti elementai su ryšiais *MSOSA* įrankyje. Toliau sukuriama PA diagramos vizualizacija, kurioje pavaizduojami skaitmenizuoti elementai. Pasirinkus visus diagramos elementus įjungiamas ryšių rodymas tarp elementų. Rekomenduojama elementus išdėstyti pagal pradinį eskizą, kad būtų paprasčiau patikrinti, ar viskas teisingai skaitmenizuota, arba pasirinkti automatinį simbolių išdėstymą. Toliau atliekami reikalingi diagramos pataisymai, pavyzdžiui, pataisomi praleisti, neteisingi pavadinimai. Jeigu reikia pakeičiamas ryšio tipas arba kryptis bei kiti reikalingi pataisymai atkuriant diagramą pagal pradinį eskizą.

4.7.2. Sprendimo trūkumai ir tobulinimas

4.7.2.1. Elementų pozicijos išlaikymas pagal pradinį eskizą

Šiuo metu nėra realizuotas skaitmenizuotų diagramos elementų pozicijų išlaikymas pagal pradinį eskizą, todėl skaitmenizavus diagramą rankiniu būdu reikia išdėstyti simbolius, arba naudoti automatinį simbolių išdėstymą. Pačioje XMI specifikacijoje nėra apibrėžta dėl elementų vizualizavimo, pavyzdžiui, jų padėties diagramose koordinačių nurodymo. XMI skirtas modelių struktūrai, semantikai ir metaduomenims pateikti standartizuotu XML formatu. Jame siekiama užfiksuoti pagrindinę informaciją apie elementus, jų ryšius, savybes ir kitas svarbias detales, o ne jų vizualinį atvaizdavimą. Vizualinį elementų išdėstymą diagramose paprastai tvarko modeliavimo priemonės arba programinė įranga, interpretuojanti XMI duomenis. Taigi, nors diagramų vizualizavimas neįtrauktas į XMI specifikaciją, tobulinant įrankį būtų galimą realizuoti elementų pozicijų išlaikymą pagal pradinį eskizą atsižvelgus į tai, kaip vizualizavimą yra realizavusios specifinės CASE tipo sistemos, pavyzdžiui, MSOSA.

4.7.2.2. Diagramos elementų atpažinimo tikslumo gerinimas

Norint pagerinti įrankio skaitmenizavimo tikslumą, pirmiausia reikia pagerinti diagramos elementų atpažinimo tikslumą. Nors diagramos elementai yra atpažįstami ~0.96 atkūrimu (4.3 lentelė), preciziškumas siekia tik ~0.91. Žemas preciziškumas yra gaunamas dėl naudojamo mažo patikimumo ir reiškia, kad reikalingas papildomas atpažintų elementų apdorojimas, kad būtų atmesti pertekliniai elementai, arba neteisingai atpažinti elementai. Norint naudoti aukštesnį atpažinimo patikimumą ir pasiekti aukštesnį preciziškumą, reikalingas išsamesnis duomenų rinkinys, kuriame būtų kuo daugiau įvairių tam tikro elemento variacijų, skirtingi diagramų fonai, elementų dydžiai, pasukimai, skirtingų stilių diagramos, nubraižytos ranka arba su įrankiu. Pavyzdžiui, sudarytame pradiniame diagramų rinkinyje (3.1 lentelė) buvo mažai tiesių ryšių, todėl apmokinus diagramos simbolių atpažinimo modelį, pastarasis neatpažindavo tokio tipo ryšių. Kad būtų atpažinti tiesūs ryšiai, buvo praplėstas duomenų rinkinys su atitinkamomis diagramomis ir elementais. Įrankis gali būti ženkliai patobulintas įvedus periodinį diagramos simbolių atpažinimo modelio apmokinimą su naujomis diagramomis. Naujos diagramos gali būti dalinai suanotuotos su pačiu įrankiu, ir jeigu reikia anotacijos gali būti pataisomos ranka.

4.7.2.3. Raktinių ryšio nustatymo komponento tobulinimas

Raktinių taškų nustatymo komponentas yra vienas iš svarbiausių įrankio komponentų, nuo kurio priklauso diagramos simbolių sujungimo atkūrimas, kuris šiuo metu yra 0.84. Šis komponentas veikia tikrinant priekinio plano (angl. *foreground)* pikselių kiekį kampuose. Nors toks primityvus būdas leidžia pasiekti ganėtinai aukštų rezultatų, norint skaitmenizuoti sudėtingesnes diagramas, pavyzdžiui, su persidengiančiais ryčiais arba ryšiais sudarytais iš keletos kreivių, reikalingi sudėtingesni metodai. Vienas iš galimų šio komponento patobulinimų būtų sukurti atskirą ryšio rodyklių pozicijų nustatymo modelį. Ryšio raktinių taškų regionų nustatymas pagal priekinio plano pikselius gali nesuveikti teisingai, jeigu yra persidengiančių teksto ar kito tipo elementų regionų ar ryšio regiono kampų.

Išvados

1. Nors šiuo metu rinkoje egzistuoja įrankių leidžiančių projektuoti UML panaudojimo atvejų diagramas, praktikoje pirmieji diagramų eskizai gali būti nubraižomi ranka ant išmaniosios lentos arba naudojant įrankį palaikantį panašią į UML notaciją. Norint efektyviai pasidalinti eskizu su suinteresuotais asmenimis bei toliau plėtoti pradinį diagramos eskizą, pastarąjį reikia rankiniu būdu perkelti į UML standartą palaikantį įrankį. Rankinis diagramos perkėlimas yra neefektyvus, tačiau nėra įrankio, kuris leistų automatizuoti UML panaudojimo atvejų diagramos skaitmenizavimą iš eskizo.
2. Ranka braižytų diagramų atpažinimo metodai literatūroje skirstomi į dvi pagrindines kategorijas: realaus laiko braižomų diagramų (angl. *online-targeted*) ir eskizų (angl. *offline-targeted)* atpažinimą ir skaitmenizavimą. Realaus laiko braižomų diagramų atpažinimo metodai negali būti sėkmingai pritaikyti diagramų eskizų skaitmenizavimui, dėl tokių metodų priklausomybės nuo istorinių diagramos braižymo duomenų, kurie eskizuose neegzistuoja. Diagramų eskizų skaitmenizavimui dažniausiai naudojami CNN mašininio mokymosi metodai. Verta paminėti, kad analizuoti sprendimai negali atlikti pilno diagramos skaitmenizavimo ir eksportavimo, nes nėra skaitmenizuojamas tekstas. Atlikus teksto atpažinimo (OCR) galimybių analizę, buvo nuspręsta naudoti *Google Cloud Vision OCR* paslaugą.
3. Egzistuojančios diagramų skaitmenizavimo sąsajos siekia aukštus diagramos simbolių atpažinimo rezultatus (98.4% - 99.5%.), tačiau sprendimai nėra išplėtoti ir prieinami galutiniam naudotojui. Sprendimai yra pritaikyti standartizuotų srautų (angl. *flowchart*) ir būsenų (angl. *finite automata*) diagramų skaitmenizavimui, bet nepalaiko UML panaudojimo atvejų (angl. *Use Case*) diagramos atpažinimo galimybių, diagramos eksportavimo į XMI formatą bei nesuteikia įrankio, leidžiančio atlikti visą diagramos eskizo skaitmenizavimo procesą galutiniam naudotojui.
4. Atlikus sistemos projektavimą buvo nuspręsta sprendimą išskaidyti į 5 pagrindinius komponentus: diagramos simbolių atpažinimo komponentą, ryšių raktinių taškų nustatymo komponentą, OCR komponentą, simbolių susiejimo komponentą ir diagramos konvertavimo į XMI failą komponentą. Toks sistemos išskaidymas suteikia galimybes realizuoti sistemą atskirais etapais; tobulinti atskirus komponentus nepriklausomai vienas nuo kito; lengviau ištestuoti sistemą; pritaikyti tinkamiausias technologijas skirtingų komponentų realizacijoms.
5. Sėkmingai realizuoti diagramos simbolių atpažinimo, ryšių raktinių taškų nustatymo, OCR, simbolių susiejimo ir diagramos konvertavimo į XMI failą komponentai. Diagramos simbolių atpažinimo mašininio mokymosi modeliui buvo panaudotas mokymosi perkėlimas (angl. *transfer learning*) ir egzistuojantis *Faster R-CNN ResNet152 V1 800x1333* modelis buvo papildomai apmokintas su 346 PA diagramų rinkiniu.
6. Įrankiui patikrinti buvo atlikti du eksperimentai: diagramos simbolių atpažinimo ir diagramos skaitmenizavimo eksperimentai su 32 naujų PA diagramų. Atlikus pirmąjį eksperimentą buvo gautas bendras visų elementų atpažinimo atkūrimas 0.96, o preciziškumas – 0.91. F1 statistikos įvertinimas yra 0.94. Antrasis eksperimentas buvo atliktas siekiant įvertinti papildomo apdorojimo žingsnių efektyvumą, XMI generavimo ir elementų sujungimo tikslumą. Bendras visų elementų skaitmenizavimo atkūrimas yra 0.93, preciziškumas 0.97, o F1 statistikos įvertinimas – 0.95. Diagramos elementų sujungimo atkūrimo įvertinimas yra 0.84, preciziškumas 0.95, o F1 statistikos įvertis – 0.89.

Literatūros sąrašas

1. OBJECT MANAGEMENT GROUP. *UML 2.5.1 specification* [interaktyvus]. 2017 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://www.omg.org/spec/UML/2.5.1/PDF

2. PETRE, Marian. UML in practice. *2013 35th International Conference on Software Engineering (ICSE)*. San Francisco, CA, USA: IEEE, 2013, p. 722-731.

3. FANG, Jiaqi, FENG, Zhen ir CAI, Bai. DrawnNet: Offline Hand-Drawn Diagram Recognition Based on Keypoint Prediction of Aggregating Geometric Characteristics. *Entropy (Basel)* [interaktyvus]. 2022, 24(3), 425 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.3390/e24030425

4. BRESLER, Martin, VACLAV, Hlavac, DANIEL, Prusa. Modeling Flowchart Structure Recognition as a Max-Sum Problem. *2013 12th International Conference on Document Analysis and Recognition*. Washington, DC, USA: IEEE, 2013, p. 1215-1219.

5. HE, Kaiming, GKIOXARI, Georgia, ir kiti. Mask R-CNN. *arXiv.org* [interaktyvus]. March 2017 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.48550/arXiv.1703.06870.

6. BRESLER, Martin, TRUYEN, V.P, ir kiti. Recognition System for On-Line Sketched Diagrams. *2014 14th International Conference on Frontiers in Handwriting Recognition.* Hersonissos, Greece: IEEE, 2014, p. 563 – 568.

7. HO-QUANG, Truong, R.V. CHAUDRON, Michel ir kiti. Automatic Classification of UML Class Diagrams from Images. *2014 21st Asia-Pacific Software Engineering Conference*. Jeju, Korea (South): IEEE, 2014, p. 399 – 406.

8. BRESLER, Martin, PRUSA, Daniel, HLAVAC, Vaclac. Online recognition of sketched arrow-connected diagrams. *IJDAR* [interaktyvus]. 2016, 19, 253 – 267 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1007/s10032-016-0269-z.

9. SCHAFER, Bernhard, LEOPOLD, Henrik, ir kiti. Sketch2BPMN: Automatic Recognition of Hand-Drawn BPMN Models. *33rd International Conference, CAiSE 2021*. Melbourne, VIC, Australia: Springer, Charm, 2021, p. 344 – 360.

10. LANO, Kevin. *UML 2 Semantics and Applications*. John Wiley & Sons, Inc., 2012. ISBN 9780470522622.

11. VISUAL PARADIGM. What is Unified Modelling Language (UML) [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/.

12. KO, Minhyuk, SEO, Yong – Jin, ir kiti. Extending UML Meta-model for Android Application. *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*. Shanghai, China: IEEE, 2012, p. 669 – 674.

13. W3C. Extensible Markup Language (XML). *W3C* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://www.w3.org/TR/xml/.

14. OBJECT MANAGEMENT GROUP. XMI Mapping Specification, v2.4.1. *OMG* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://www.omg.org/spec/XMI/2.4.1/PDF.

15. WIKIPEDIA. List of Unified Modeling Language tools. *Wikipedia* [interaktyvus]. N.d [žiūrėta 2024 m. gegužė 27 d.]. Prieiga per internetą: https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools.

16. BOSER, E. Bernhard, GUYON, Isabelle, ir kiti. A training algorithm for optimal margin classifiers. *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, p. 144 – 152.

17. PASUPA, Kitsuchart, SUNHEM, Wisuwat. A comparison between shallow and deep architecture classifiers on small dataset. *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE).* Yogyakarta, Indonesia: IEEE, 2016, p. 1 – 6.

18. WU, Qiang, ZHOU, Ding – Xuan. SVM Soft Margin Classifiers: Linear Programming versus Quadratic Programming. *Neural Computation* [interaktyvus]. 2005, 17(5), 1160 – 1187 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1162/0899766053491896

19. PAN, Chen, YAN, Xiangguo ir kiti. Hard Margin SVM for Biomedical Image Segmentation. *Advances in Neural Networks – ISNN*. Berlin, Heidelber: Springer, Berlin, 2005, p. 754 – 759.

20. CM, Salgado, Azevedo, C. Noise Versus Outliers. Secondary Analysis of Electronic Health Records. *Springer Cham* [interaktyvus]. 2016, 14 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1007/978-3-319-43742-2_14

21. KEERTHI, Sathiya, LIN, Chih – Jen. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation* [interaktyvus]. 2003, 15(7), 1667 – 1689 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1162/089976603321891855.

22. CORTES, Corinna, VAPNIK, Vladimir. Support – vector networks. *Springer* [interaktyvus]. 1995, 20, 273 – 297 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1007/BF00994018

23. THARWAT, Alaa. Parameter investigation of support vector machine classifier with kernel functions. *Springer* [interaktyvus]. 2019, 61, 1269 – 1302 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1007/s10115-019-01335-4

24. XUN, Wang, YU, Wei, ir kiti. Detecting Worms via Mining Dynamic Program Execution. *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*. Nice, France: IEEE, 2007, p. 412 – 421.

25. PATLE, Arti, SING, Deepak. SVM kernel functions for classification. *2013 International Conference on Advances in Technology and Engineering (ICATE)*. Mumbai, India: IEEE, 2013, p. 1 – 9.

26. AMARAL, Igor, COSTA, Joaquim. Hierarchical medical image annotation using SVM – based approaches. *Conference on Information Technology and Applications in Biomedicine.* Corfu, Greece: IEEE, 2010, p. 1 – 5.

27. LI, X, WANG, L. Multilabel SVM active learning for image classification. *2004 International Conference on Image Processing, 2004.* Singapore: IEEE, 2004, p 2207 – 2210.

28. WIKIPEDIA. Artificial neural network. *Wikipedia* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://en.wikipedia.org/wiki/Artificial_neural_network.

29. HIJAZI, Samer, KUMAR, Rishi ir kiti. Using Convolutional Neural Networks for Image Recognition. *Cadence*. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://site.eet-china.com/webinar/pdf/Cadence_0425_webinar_WP.pdf

30. ALBASHISH, Dheeb, SAYYED-AI, Rizik. Deep CNN Model based on VGG16 for Breast Cancer Classification. *2021 International Conference on Information Technology (ICIT)*. Amman, Jordan: IEEE, 2021, p. 805 – 810.

31. LECUN, Y, BOTTOU, L ir kiti. Gradient-based learning applied to document recognition. *Proceedings of the IEE* [interaktyvus]. 1998, 86(11), p. 2278 – 2324 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1109/5.726791

32. KRIZHEVSKY, Alex, SUTSKEVER, Ilya. ImageNet classification with deep convolutional neural networks. *Communicationsof the ACM* [interaktyvus]. 2017, 60(6), p. 84 – 90 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1145/3065386

33. OVTCHAROV, Kalin, RUWASE, Olatunji ir kiti. Accelerating Deep Convolutional Neural Networks Using Specialized Hardware. *Microsoft Research.* 2015. [žiūrėta 2024 m. gegužės 27 d.].

34. CHAPLE, Girish, DARUWALA, R.D. Design of Sobel operator based image edge detection algorithm on FPGA. *International Conference on Communications and Signal Processing.* Melmaruvathur, India: IEEE, 2014, p. 788 – 792.

35. YINGEE, Huo, ALI, Imran irkiti. Deep Neural Networks on Chip - A Survey. *International Conference on Big Data and Smart Computing (BIGCOMP).* Busan, Korea (South): IEEE, 2020, p. 589 – 592.

36. HU, Zheng, ZHANG, Jiaojiao. Handling Vanishing Gradient Problem Using Artificial Derivative. *IEEE Acccess* [interaktyvus]. 2021, 9, p. 22371 – 22377 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per doi: 10.1109/ACCESS.2021.3054915

37. KIM, Bubryur, YUVARAJ, N. Surface crack detection using deep learning with shallow CNN architecture for enhanced computation. *Neural Computing and Applications* [interaktyvus]. 2021, 33, p. 9289 – 9350 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per doi: 10.1007/s00521-021-05690-8

38. KAYED, Mohammed, ANTER, Ahmed ir kiti. Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture. *International Conference on Innovative Trends in Computer Engineering (ITCE).* Aswan, Egypt: IEEE, 2020, p. 238 – 243.

39. BROCK, Andrew, DE, Soham ir kiti. High – Performance Large – Scale Image Recognition Without Normalization. *arXiv.org* [interaktyvus]. March 2021 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.48550/arXiv.2102.06171

40. SABU, Abin, DAS, Anto Sahaya. A Survey on Optical Character Recognition System. *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*. Tiruchengode, India: IEEE, 2018, p. 152 – 155.

41. PRAHBU, P. Digital Image Processing Techniques-A Survey. *International Multidisciplinary Research Journal* [interaktyvus]. 2016, 5(11) [žiūrėta 2024 m. gegužės 27 d.]. ISSN No.2231-5063.

42. MITTAL, Rishabh, GARG, Anchal.Text extraction using OCR: A Systematic Review. *International Conference on Inventive Research in Computing Applications (ICIRCA).* Coimbatore, India: IEEE, 2020, p. 357 – 362.

43. H. F. Schantz, The history of OCR, optical character recognition, [Manchester Center, Vt.] : Recognition Technologies Users Association, 1982, p. 114.

44. HAMAD, Karez, KAYA, Mehmet. A Detailed Analysis of Optical Character Recognition Technology. *International Journal of Applied Mathematics Electronics and Computers. 2016, p. 244 - 249.*

45. MAINI, Raman, AGGARWAL, Himashu. A Comprehensive Review of Image Enhancement Techniques. *arXiv.org* [interaktyvus]. March 2010 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.48550/arXiv.1003.4053

46. OPENCV. Smoothing Images. *OpenCv* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html.

47. OPENCV. Image Tresholding. *OpenCv* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.

48. SHINDE, Archana, CHOUGULE, D.G. Text Pre-processing and Text Segmentation for OCR. *International Journal of Computer Science Engineering and Technology* [interaktyvus]. January 2012, vol 2, no. 1, 810 – 812 [žiūrėta 2024 m. gegužės 27 d.]. ISSN:2231-0711.

49. CHOUGULE, D.G. Text Pre-processing and Text Segmentation for OCR. *IJCSET* [interaktyvus]. January 2012, vol 2[žiūrėta 2024 m. gegužės 27 d.]. ISSN:2231-0711.

50. HE, Xiangjian, ZHENG, Lihong ir kiti. Segmentation of characters on car license plates. *IEEE Workshop on Multimedia Signal Processing.*Cairns, QLD, Australia: IEEE, 2008, 399-402.

51. CLOUDHARY, Amit, RISHI, Rahul ir kiti. A New Characted Segmentation Approach for Off-Line Cursive Handwritten Words. *Elsevier* [interaktyvus]. 2013, 17, p. 88 – 95 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per doi: 10.1016/j.procs.2013.05.013

52. PRASAD, Vijay, JAYANTA, Yumnam. A study on method of feature extraction for Handwritten Character Recognition. *IJST.* 2013, 6(3), p. 174 – 178. [žiūrėta 2024 m. gegužės 27 d.].

53. KUMAR, Munish, JINDAL, M. K. ir kiti. Offline Handwritten Gurmukhi Character Recognition: A Review. *Springer Link* [interaktyvus]. 2016, 87, p. 137 – 143 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1007/s40010-016-0284-y

54. JOSE, I. KNN (K-Nearest Neighbors) #1. *Towards Data Science* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d.

55. SARIKA, Naragudem, SIRISALA, Nageswararao. CNN based Optical Character Recognition and Applications. *International Conference on Inventive Computation Technologies (ICICT).* Coimbatore, India: IEEE, 2021, p. 666 – 672.

56. THAKUR, R. Step by step VGG16 implementation in Keras for beginners. *Towards Data Science* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c.

57. JATOWT, Adam, DOUCET, Antoine. Survey of Post-OCR Processing Approaches. *ACM Computing Surveys* [interaktyvus]. 2021, 124, p. 1 – 37 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1145/3453476

58. TESSERACT-OCR. Tesseract-OCR. *GitHub.* N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://github.com/tesseract-ocr/tesseract.

59. AMAZON. Amazon Textract. *Amazon.* N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://aws.amazon.com/textract/.

60. GOOGLE. *Google Cloud Vision API. Google* [interaktyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://cloud.google.com/vision/?utm_source=google&utm_medium=cpc&utm_campaign=emea-emea-all-en-dr-bkws-all-all-trial-e-gcp-1707574&utm_content=text-ad-none-any-DEV_c-CRE_683761577022-ADGP_Hybrid+%7C+BKWS+-+EXA+%7C+Txt+-+AI+And+Machine+Learning+-+Vision+A.

61. DILMEGANI, C. OCR in 2024: Benchmarking Text Extraction/Capture Accuracy. *AiMultiple.* N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://research.aimultiple.com/ocr-accuracy/.

62. SCHAFER, Bernhard, KEUPER, Margret ir kiti. Arrow R-CNN for handwritten diagram recognition. *IJDAR. Springer Link* [interaktyvus]. 2021 24, p. 3 – 17 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.1007/s10032-020-00361-1

63. REFAAT, Khaled, HELMY, Wael ir kiti. A new approach for context-independent handwritten offline diagram recognition using support vector machines. *International Joint Conference on Neural Networks (IJCNN).* Hong Kong: IEEE, 2008, p. 177 – 182.

64. REN, Shaoqing, HE, Kaiming ir kiti. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv.org* [interaktyvus]. March 2015 [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per: doi: 10.48550/arXiv.1506.01497

65. TENSORFLOW. TensorFlow 2 Detection Model Zoo. *Tensorflow* [intekratyvus]. N.d [žiūrėta 2024 m. gegužės 27 d.]. Prieiga per internetą: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md.