Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство
Ниво студија: Основне академске студије
Модул: Информатика у инжењерству
Предмет: Базе података
Број индекса: 114/2014

Лука Б. Стојковић

Пројектовање база података применом UML-а

завршни рад

Комисија за преглед и одбрану:

1. **Проф. др Милан Ерић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

Изјава о самосталној изради рада

Изјављујем да сам овај завршни рад урадио самостално, служећи се стеченим знањем и искуством као и наведеном литературом.

114/2014 Лука Стојковић

_____

(потпис)

У оквиру завршног рада са темом "*Пројектовање база података применом UML-а*" кандидат треба да:

Проучи литературу и презентира концепт примене UML-а у пројектовању релационих база података. Рад треба да обухвати уводна разматрања везана за развој и примену UML-а, опис елемената UML-а као и опис дијаграма које подржава UML. На реалном примеру приказати примену UML-а. На крају дати закључна разматрања и списак коришћене литературе.

Препоручена литература:

1. Полишчук Ј.: **Пројектовање информационих система**, Електротехнички факултет, Подгорица, 2007.
2. Rainer K., Turban E.: **Introduction to Information Systems**, John Wiley & Sons, Inc., 2009
3. Лазаревић Б., ……: **Базе података**, ФОН Београд, Београд 2003.
4. Вељовић А.: **Основе објектног моделирања - UML**, Компјутер библиотека, 2012.
5. Naiburg E., Maksimchuk R.: **UML for Database Design**, Addison-Wesley, Boston, 2012.

Крагујевац, јун 2017.

Ментор:

Prof. Dr Milan D. Erić

Резиме

У овом раду говори се о UML-у, језику за моделирање база података који је због своје једноставности и функционалности постао стандард. Описана је намена UML дијаграма као и њихове основне карактеристике. Приказане су функције и интеракције компонената од којих је дијаграм сачињен. Ради бољег објашњења како дијаграми функционишу, дат је пример реалног система са детаљним објашњењем сваког дијаграма.

Кључне речи: UML, дијаграми, моделирање, класе, објекти, интеракција између компоненти, функционалност.

Summary

This labor discusses about UML, a database modeling language that has become standard because of its simplicity and functionality. This describes the purpose of UML diagrams and their basic characteristics. The functions and interactions of the components from which the diagram is compiled are shown here. For a better explanation of how the diagram works, it is an example of a real system with a detailed explanation of each diagram.

Key words: UML, diagrams, modeling, class, objects, interaction between components, functionality

Садржај

1. Увод

Базе података у општем смислу представљају скуп података смештених на једном месту. Базе података постоје веома дуго, некада је то било у писаним формама а данас се оне налазе у електронском облику, у сваком случају имају исту намену а то је да прикупљају одређене податке.

Постоје различите технике које се користе за прављење структуре модела база података. Структура већине модела састоји се је од једнеог главном модела, технички речено, ти модели представљају усавршени основни модел за управљање базама података.

По неким изучавањима, базе података се могу посматрати из два угла. Из једног угла се посматрају као системи за управљање базама података, док се из другог угла оне посматрају као модели података.

- Под системом за управљање базама података мисли се на софтвер који врши обраду података, односно, софтвер који врши једноставно претраживање података, вишеструко паралелно коришћење истих података као и поузданост и сигурност тих података. [1]
- Модели података представљају специфичне теорије помоћу којих се спецификује и пројектује нека конкретна база података или информациони систем. [1]

2. Настанак UML-а

Развојни методи за старе програмске језике као што су Fortran и Cobol развили су се раних 70-тих година. У то време, углавом су се користили структурна анализа, структурни дизајн и структурни дизајн у реалном времену. Ови програмски језици су пре свега настали за потребе војске, што је значило да организација мора бити на веома високом нивоу услед чега је уследило да и само документовање система и његовог развоја такође буде на високом нивоу. Након војске, ови програмски језицу су такође били коришћени и за потребе различитих компанија. Бизнисмени нису били задовољни са тим што је сложеност њихове апликације беспотребно била на високом нивоу, што је условило да свака компанија која поседује апликацију, мора да плаћа услуге специјализованих компанија за надзор саме апликације. Такође су сматрали да је за њихове потребе то могло бити много једноставније урађено, самим тим, услуге компанија за надзор апликације не би биле потребне чиме би и цена саме апликације била много мања.

Званично, први објектно-оријентисани програмски језик јесте Simula која је настала 1967. године. За овим програмским језиком настали су и други као што су Objective-C, C++, Eiffel и CLOS. Сваки од ових програмских језика имао је сложену терминологију, дефиницију и нотацију, такође, језици су се међусобно доста разликовали. Ова сложеност и разноликост језика је утицала на то да се корисници теже упусте у учење и разумевање истих.

Управо због ових проблема, било је јасно да је неопходно доћи до неког споразума у вези терминологије и технике којима би се објектно-оријентисане идеје представиле.

Године 1997, заједничким радом Rumbaugh-a, Booch-a i Jackobson-a креиран је UML који тада постаје нека врста стандарда објектно-оријентисаног моделирања.

3. UML (Unified Modeling Language)

Брз развој хардвера и софтвера као и све веће потребе да се до жељених информација стигне у што краћем временском периоду, условиле су настанак новог језика за моделирање база података.

UML је стандардизовани графички језик за пројектовање, спецификацију, визуелизацију и документацију софтверских система. Први разлог за његово коришћење јесте богата нотација за моделирање која је независна од процеса моделирања, а други разлог је то што сам језик може да се фокусира, односно да се специјализује за само једну област.[3]

UML је како и само име каже, јединствени језик за моделирање. Смисао овог језика јесте да сви тимови који раде на креирању неке базе података, користе исти језик ради боље међусобне комуникације. Такође је било потребно креирати неки језик који ће и самом кориснику бити доступан, односно разумљив.

Може се рећи да је UML настао као комбинација свих језика за управљање базама података из којих је извучено само оно најбоље и као такав постао је стандард.

UML је пре свега језик који се користи за објектно оријентисано моделирање. Како је реч о објектно оријентисаном моделирању, то значи да је овај вид моделирања заснован на чињеници да систем представља скуп међусобно повезаних објеаката, где се стање система посматра као као стање објекта, а операције над објектима се посматрају као реализација функција.

UML чува два типа информација. Први тип информација се заснива на статичкој структури, док се други тип заснива на динамичком понашању система. Систем представља целину која се састоји од различитих компонената и као резултат, он даје повратне информације које су корисне за самог корисника.

Статичка структура дефинише особине објекта који су важни за систем и његову имплементацију, као и постојање везе између ових објеката.[3]

Динамичка компонента UML-а има за циљ да јасно опише какве се промене дешавају објектима у одређеном временском интервалу док систем ради, као и који објекти међусобно комуницирају за то време и какве су њихове поруке.

Објекти нису ништа друго него ентитети који су способни да чувају своја стања и који стављају на располагање околни скуп операција преко којих се та стања приказују или мењају. [2]

Стање објекта представља скуп информација о његовој прошлости и садашњости чиме се дефинише његово будуће понашање под дејством дефинисаног скупа његових операција. [2]

Објектно оријентисан развој информационих система значи да се проблем сагледава преко објеката који представљају људе, раднике, предузећа, градове, све оне ствари које се користе у датој бази података.

4. UML-креирање модела

За опис рада у пословним системима се не могу користити природни језици јер постоје речи које имају двосмислена значења која би доводила до збуњивања приликом моделирања неког система. Такође се не може користити ни формални језик јер је он неразумљив за већину људи. Из ових проблема се развио поступак моделирања који се показао као најуспешнија техника за раумевање, односно, управо овај поступак омогућава једноставну комуникацију између програмера и корисника.

Како је идеја о моделирању као представљању једноставије реалности прихваћена, то значи да управо то моделирање представља развој система.

Када је потребно направити неки комплекснији пројекат, то такође значи да је потребно извршити и комплексније моделирање. Постоје многи фактори који утичу на успешност пројекта, али постојање прецизног језика за моделирање је један од најбитнијх.

Језик за моделирање мора да укључује:

- Елементе моделирања (основни принципи моделирања и семантика)
- Нотација (визуелно тумачење елеманта моделирања) и
- Језичке изразе који се користе у самој реализацији. [2]

Моделирањем се постижу четири ствари:

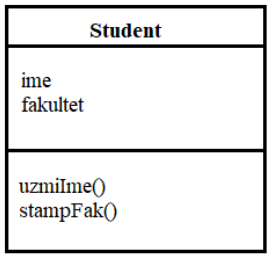
- Модели помажу да се систем визуелизује онакав какав је или какав би требало да буде
- Модели омогућују да се одреди структура или понашање система
- Модели дају узорке који нас воде приликом конструкције система
- Модели документују одлуке које смо доносили. [2]

Једноставније речено, UML се не заснива ни на једном програмском језику за развој објектно оријентисаних система, већ он предстаљва општи модел дизајна.

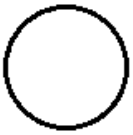
5. Компоненте UML-а

Сваки UML дијаграм сачињен је од одређених графичких симбола који су међусобно повезани. Поред тих симбола може, међутим није нужно, стајати неко текстуално објашњење.

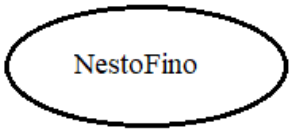
- Блок за представљање класа и елемената класе



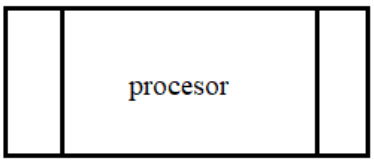
- Симбол за престављање интерфејса



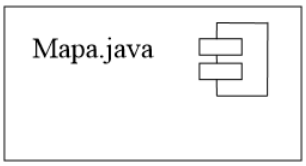
- Блок за представљање корисничких функција помоћу којих се описују понашања у моделу



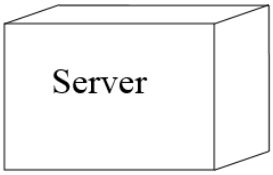
- Симбол за представљање активне класе



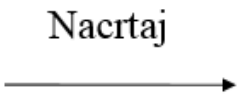
- Симбол за представљање компоненти



- Симбол за представљање чвора физичког елемента који представљају неки рачунарски „resurs“



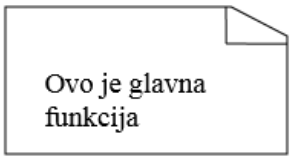
- Симбол помоћу којег се приказује интеракција, односно размена порука између објеката.



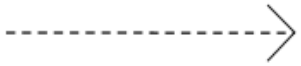
- Симбол помоћу којег се приказује неко стање

- Симбол који представља груписање неких елемената

- Симбол за приказивање коментара



- Зависност између неких елемената

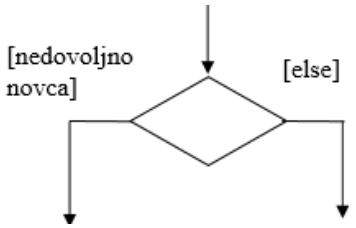


- Кардиналност између објеката

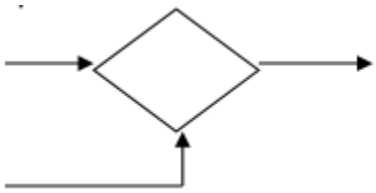
- Симбол за представљање генерализације

- Симбол који представља реализацију

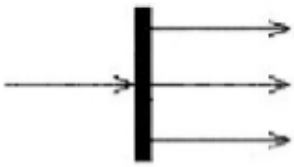
- Гранање



- Спајање



- Иницијализација паралелних процеса



6. Дијаграми

Дијаграм представља графички приказ скупа међусобно повезаних елемената. Дијаграми се најчешће представљају у облику делова повезаних гранама, односно релацијама. У суштини, дијаграм представља једноставнији, илустровани приказ организације неког система. Током развоја софтвера, појављивали су се многи системи који су захтевали коришћење дијаграма, сходно томе, за различите системе били су потребни различити типови дијаграма за решавање одређених проблема. Најосновнија подела дијаграма у UML-у јесте на:

- Структурне дијаграме
- Дијаграме понашања
- Дијаграме интеракције

6.1. Структурни дијаграми

Моделирање компоненти система се врши помоћу структурних дијаграма. Структурни дијаграми се деле на:

- Дијаграме класа
- Дијаграме компоненти
- Дијаграм сложене структуре
- Дијаграм распоређивања
- Дијаграм објекта
- Дијаграм пакета[4]

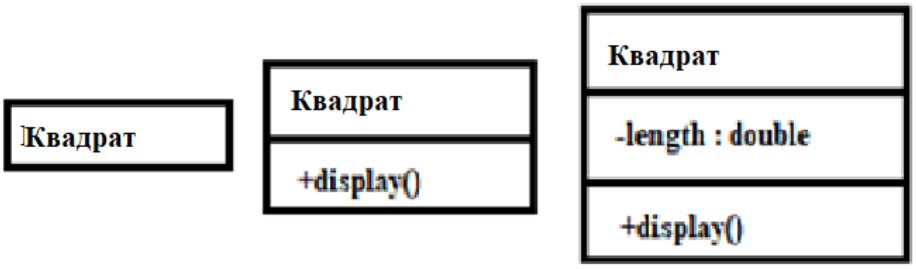
6.1.1. Дијаграми класа

Основни UML дијаграм је дијаграм класе. Он описује класе и приказује везу између њих. Типови могућих веза су:

- Када је једна класа „врста“ друге класе: веза „**јесте**“
- Када постоје везе између две класе
    - Једна класа „садржи“ другу класу: веза „**има**“
    - Једна класа „**користи**“ другу класу

На следећој слици дат је пример класа, односно, сваки правугаоник представља класу. У свакој класи могу да се представе:

- Име класе
- Подаци чланова класе
- Методе класе



Слика 1. Дијаграм класе

Приказивање класа се може представити на три начина. Према слици 1. начини представљања класа су следећи:

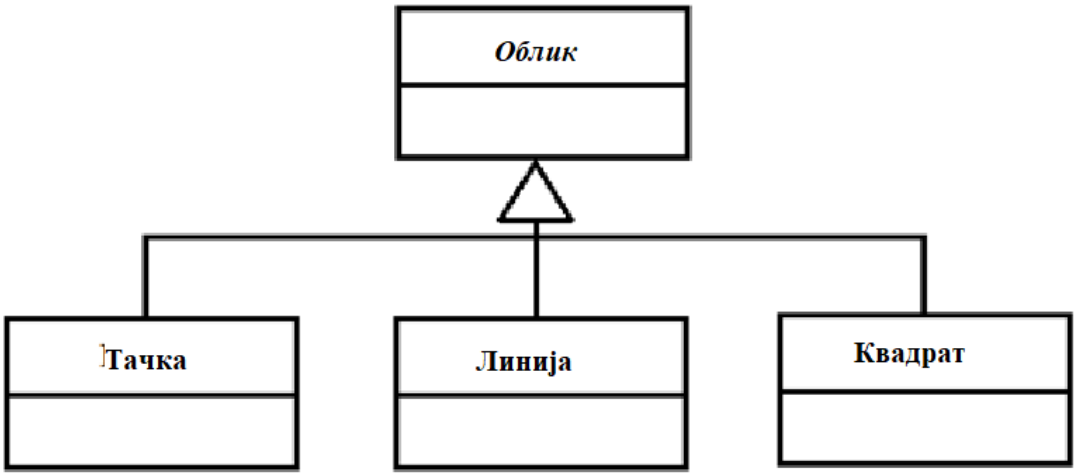
- Име класе (први начин)
- Име и метод класе (други начин)
- Име и метод класе уз податке чланова калсе (трећи начин)

Први начин приказивања је представљен у левом правоугаонику који представља име класе и ништа више. Такав начин приказивања класа се користи само у случајевима када није потребно детаљније описати дату класу, односно нису потребне информације о тој класи.

Други начин приказивања класе је представљен у средњем правоугаонику који поред имена класе представља још и њен метод. Као што је већ напоменуто, класа чије је име *Квадрат* садржи методу *display*. Знак + испред речи *display* (која представља назив методе класе) нам говори о томе да је метода јавна и да је објекти друге класе могу позвати.

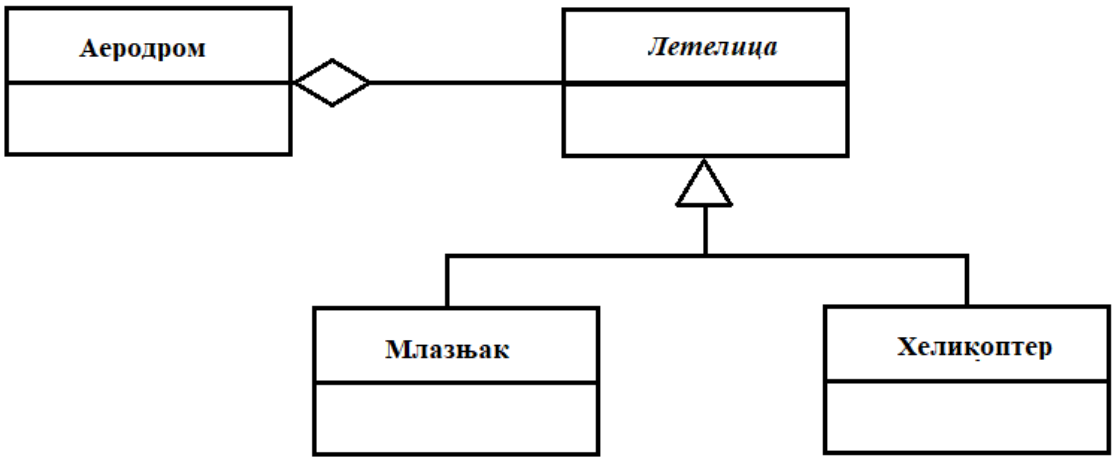
Трећи начин приказивања класе је у суштини веома сличан другом начину, односно приказује име и методу класе али, у овом начину приказивања додати су још и подаци чланова класе. У овон начину приказивања знак – пре атрибута *length* показује да је вредност атрибута приватна, а то би значило да је вредност тог атрибута доступна само објекту којем припада.

Како је већ напоменуто, дијаграми класа служе за приказ веза између различитих класа.



Слика 2. Дијаграм класе који приказује везу „јесте“

На слици 2. показана је веза класе *Shape* и неколико класа које су из ње изведене. Стрелица која показује на класу *Облик* говори о томе да сву све класе које показују на класу *Облик* управо изведене из ње. Само име класе *Облик* је искошено с намером. Искошено име говори о томе да је реч о апстрактној класи, односно да је реч о класи која служи да се дефинише интерфејс за класе које су из ње изведене.

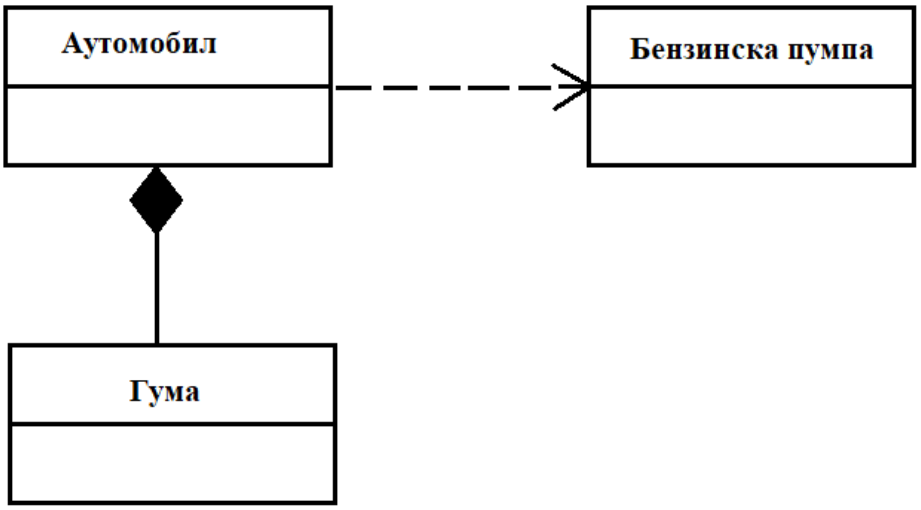


Слика 3. Дијаграм класе који приказује везу „има“

Потребно је знати да постоје две врсте везе „има“. Објекат може поседовати неки други објекат када је поседовани објекат део тог објекта, такође у овом случају објекат може поседовати други објекат иако тај поседовани објекат није део објекта који га поседује. Уколико посматрамо слику 3. то би значило да класа *Аеродром* „има“ класу *Летелица,* међутим, класа *Летелица* није део класе *Аеродром,* и даље се може рећи да класа *Аеродром* поседује класу *Летелица.* Овакав тип везе назива се **агрегација**.

Из овог дијаграма се може видети да је *Летелица* или *Млазњак* или *Хеликоптер*. Као што је већ био случај, класа чији је назив написан искошеним словима је апстрактна класа. Из овога следује да ће класа *Аеродром* поседовати једну од класа *Млазњак* и *Хеликоптер*, али и да ће их подједнако користити као објекте класе *Летелица*. Ромб који се налази са десне стране класе говори да је реч о агрегацији.

Други тип везе „има“ се остварује када је један објекат део другог и овакав тип везе се назива **композиција**.



Слика 4. Дијаграм класе који приказује композицију и везу „користи“

На слици 4. јасно се види да класа *Аутомобил* садржи део класе *Гума*. Може се рећи да је класа *Аутомобил* сачињена од објекта класе *Гума.* Потребно је приметити да је на овом дијаграму, ромб који се налази са доње стране класе *Аутомобил,* за разлику од претходних примера, потпуно испуњен. Испуњен ромб говори о томе да је реч о композицији. На дијаграму се такође види да класа *Аутомобил* користи класу *Бензинска пумпа*. Веза која је успостављена између ове две класе је такозвана веза „користи“ и она се означава испрекиданом линијом са стрелицом усмереном ка класи чији се објекти користе.

Из ових дијаграма јасно се може приметити да и агрегација и композиција подразумевају постојање неког објекта који садржи један или више других објеката. Разлика између агрегације и композиције јесте у томе што агрегација означава да су објекти које поседује неки објекат више као скуп неких независних ствари, док композиција говори о томе да је један објекат део другог објекта.

Као што је већ неколико пута речено, дијаграми класа приказују везе између класа, с тим што се агрегација и композиција некако више односе на објекте.

6.1.2. Дијакрами компоненти

Намена ових дијаграма јесте да се прикажу компоненте неког система. Компонента неког система је уствари део истог система који се може самостално испоручити крајњем кориснику. Најбитнија ствар код дијаграма компоненти јесте усаглашеност. Дијаграм компоненти је онај код којег се уношењем нове компоненте у систем не ремети рад система. Углавном су компоненте састављене из једне или више класа и као такве представљају независну целину која је помоћу сопственог интерфејса повезана са остатком система. Захваљујући овој способности дијаграма компоненти, постиже се да промене које се налазе унутар једне компоненте не стварају промене унутар читавог система.

Дијаграм компоненти се састоји од:

- Компонената
- Њихових међусобних односа
- Јавне интерфисије [4]

6.1.3. Дијаграм сложене структуре

Дијаграм сложене структуре представља везу између класа, интерфејса или компоненти у циљу описа структуре задужене за извршавање посматране функционалности. [4] Овај тип дијаграма је веома сличан дијаграмима класа с тим што је једина разлика у томе да ови дијаграми за разлику од дијаграма класа који приказују статичку структуру система кроз приказ класа са њиховим атрибутима и операцијама, ови дијаграми приказују извршну архитектуру релације, промену њених елемената и њену комуникацију са окружењем.

Дијаграми сложене структуре се састоји од:

- Сложених компоненти и њихових елемената
- Тачке интеракције са другим елементима система [4]

6.1.4. Дијаграми распоређивања

Намена ових дијаграма јесте да прикажу хардверску архитектуру система, односно приказују распоређивање делова система на физичким локацијама. Помоћу ових дијаграма се врло лако може пратити конфигурација процеса у току неког извршавања, такође се преко њих могу пратити и интеракције са компонентама које се у њима налазе. Ови дијаграми се приказују помоћу чворова (који су у виду квадрата) и релација.

6.1.5. Дијаграми објекта

Дијаграми објеката су једни од ретких дијаграма који су постојали и на самом почетку формирања UML-а, међутим тада су још увек били као неформални дијаграми. Намена ових дијаграма, како им и само име говори, јесте да прикажу објекте посматраног дела система у жељеном тренутку. Прецизније речено, намена ових дијаграма јесте за детаљније описивање структуре система и углавном се користе када дијаграми класа не дају довољно квалитетне описе. Објашњавање и илустровање скупа дијаграма класа као и представљање слика објеката и њихове везе у специфичном тренутку времена је још прецизније објашњење фунцкије дијаграма објеката.

Дијаграм објекта се састоји од:

- Назива класа
- Назива објеката
- Имена и вредности атрибута

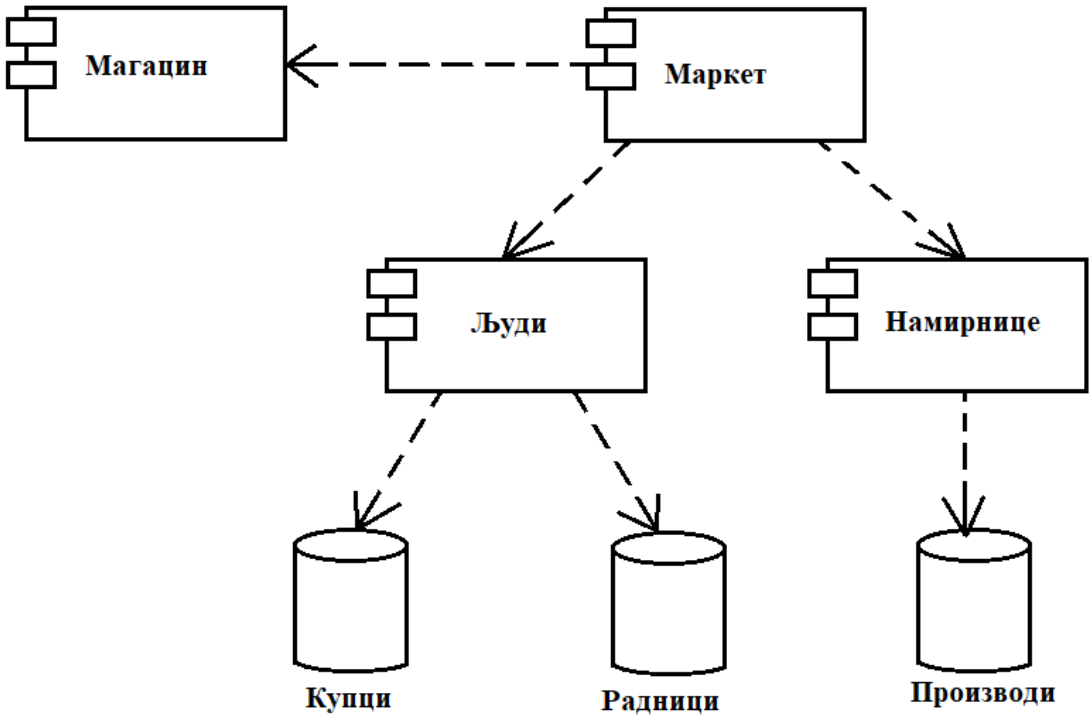
6.1.6. Дијаграми пакета

Намена пакета јесте представљање механизама за груписање UML елемената, најчешће класа. Ови пакети се такође могу користити и за груписање других елемената као што је пример груписања случајева употребе или груписање ентитета или чак и груписање самих компоненти система. Шематски, пакети се у дијаграмима приказују као проавоугаоници са језичком у горњем левом углу, а унутар правоугаоника се налази име пакета. Уколико је потребно извршити измену пакета, то би значило да ће уједно доћи и до повлачења релација зависности. Релације зависности говоре да елемти који се налазе у ралзичитим пакетима могу бити у међусобном односу, односно да између њих постоји нека зависност.

Уколико желимо да креирамо пакете калса, онда је прво потребно извршити групацију хијерархијске структуре класа или извршити груписање класа чије су инстанце у међусобној зависности при чему је та зависност приказана на дијаграмима секвенци или на дијаграмима сарадње. Одавде долази закључак да је класе могуће сместити у пакете уколико се оне налазе у логичкој или у физичкој вези и да их је као такве могуће представити као пакет на дијаграму пакета.

Дијаграм пакета се састоји од:

- Назива и граница пакета
- Класа у пакетима
- Међусобних односа класа
- Међусобних зависности класа
- Могућности коришћења у домену случајева употребе [4]



Слика 5. Пример дијаграма пакета

На слици 5 показано је како су елементи логичког модела груписани у пакете, као и међузависности самих пакета.

6.2. Дијаграми понашања

Дијаграми понашања истичу шта се дешава у систему који се моделује и дели се на три групе, а то су:

- Дијаграм активности
- Дијаграм стања
- „Use Case“ дијаграм [4]

6.2.1. Дијаграми активности

Ови дијаграми служе са истраживање и описивање логике процедура, пословних поступака и радног тока. Користе се и за приказивање акција операција у класи, слично као и традицијални дијаграми тока података. Токови функционалности који су представљени преко дијаграма случајева употребе се описују преко дијаграма активности. Дијаграми активности у суштини приказују све активности које се дешавају унат једне посматране функционалности. Један дијаграм активности има могућност да покаже један или више сценарија који се могу десити при извршавању неке фунцкионалности.

Дијаграм активности се састоји од:

- Стања активности
- Токова података
- Стања акција
- Раздвајања
- Гранања
- Спајања
- Линија аутора (може а и не мора)
- Почетну тачку
- Крајњу тачку [4]

Везу између почетне и крајње тачке чине токови података. Потребно је знати да се почетна тачка означава пуним црним кругом, док се крајња тачка означава црним кругом унутар белог круга. Стања активности и акције се обележавају правоугаоницима. Гранање за описивање услова као и тачка у којој се врши гранање се означава ромбом из којег постоје најмање два, а углавном и више токова података. Раздвајање активности се представља равном заобљеном линијом и, како и само име каже, служи за раздвајање активности када дође до већег броја активности у истом интервалу. Линија аутора се може

али и не мора користити. У сваком случају, линија аутора се користи уколико желимо да назначимо који учесник извршава коју активност.

6.2.2. Дијаграми стања

Дијаграми стања имају улогу да прикажу понашање дела система. Део система се може протумачити као објекат инстанце неке класе која се посматра. Помоћу ових дијаграма се посматрају стања посматраних објеката, транзиције измене стања као и узроке које су условиле да објекат пређе из једног стања у друго, другим речима. Овим дијаграмом су обухваћена сва стања у којима се објекат може наћи у посматраном делу времена. Дијаграми су добијени као комбинација дијаграма активности и дијаграма интеракције и углавном се користе за посматрање најзначанијих класа система.

Овим дијаграмима се у пракси најчешће моделују динамички аспекти система, односно, користе се за моделирање животног века неког објекта.

Дијаграми стања се састоје од:

- Свих могућих стања у објекту
- Посебно наглашеног почетног стања
- Посебно наглашеног стања
- Прелазака између стања [4]

6.2.3. „Use Case“ дијаграми

Ови дијаграми служе за грубо представљање функционалности посматраног система или посматраног дела организације. Ови дијаграми, или дијаграми случајева употребе, се могу поделити на два дела а то: дијаграми случајева употребе и дијаграми случајева употребе пословног процеса. [4] „Шта систем ради?“, то је главно питање на које дијаграм случајева употребе треба да да одговор, док дијаграм случајева употребе пословног процеса треба да да одговор на питање „Шта организација ради?“. Дијаграм случајева употребе има за циљ да прикаже функцију система која ће бити аутоматизована, док дијаграм случајева употребе пословног процеса има за циљ да прикаже функцију система која ће поред тога што ће бити аутоматизована још имати и мануелне функционалности.

Дијаграм случајева употребе се састоји од:

- Сценарија
- Носиоца улога (учесника)
- Интеракција

Сценарија су ништа друго него описи неких радњи, односно функција које систем обавља. Носиоци улога или учесници су корисници, односно особе, или ентитети који су вештачки створени како би учествовали у сценарију. Интеракције су радње које се обављају између два или више носиоца улога.

На на слици 6 дат је пример дијаграма случајева употребе. Тема овог дијаграма јесте да постоји купац који се ту налази као учесник у систему. Циљ купца јесте да изврши следеће радње. Како је могуће извести интеракцију између два учесника, то значи да постоји могућност увођења продавца у овај дијаграм.

Слика 6. Пример дијаграма случајева употребе

6.3. Дијаграми интеракција

Поред дијаграма класа који показују везу између класа, постоје још и дијаграми интеракција који показују везу између објеката, односно како објекти међусобно сарађују. Најучесталији тип дијаграма интеракција јесте дијаграм секвенце и врло је битно знати да се ови дијаграми читају од врха ка дну.

Дијаграми интеракције се деле на следеће:

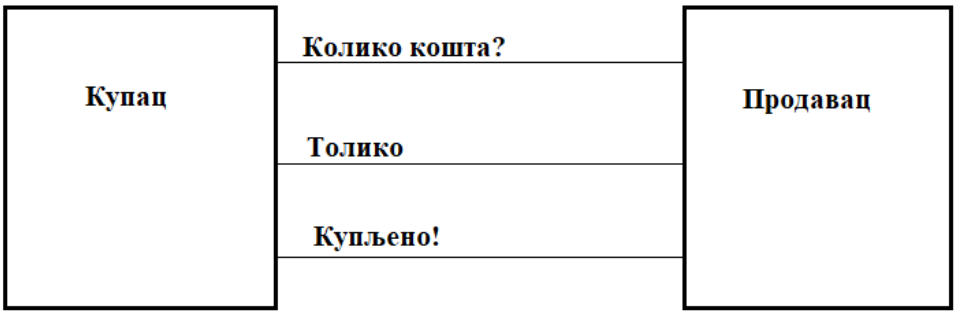
- Дијаграм сарадње
- Дијаграм прегледа интеракције
- Секвенцијални дијаграм

6.3.1. Дијаграм сарадње

Помоћу дијаграма сарадње се врши истицање структурне организације објекта који учествује у некој интеракцији. Када се погледа семантика дијаграма сарадње види се да је скоро исти као дијаграм секвенце. Оно што ова два дијаграма међусовно разликује је то што дијаграм сарадње не садржи линију живота објекта нити фокус којим се управља, а садржи објекте који су у интеракцији са везама које садрже поруке између њих.

Објекти су представљени у правоугаоницима и представљају чворове графа, а поруке које се налазе између ових објеката се представљају као гране, односно линије које се налазе између тих чворова. Ради лакшег распознавања редоследа којим су поруке послате од једног ка другом објекту, оне се могу означити редним бројевима.

На следећој слици дат је дијаграм сарадње у којем су објекти купац и продавац.



Слика 8. Пример дијаграма сарадње

6.3.2. Дијаграм секвенце

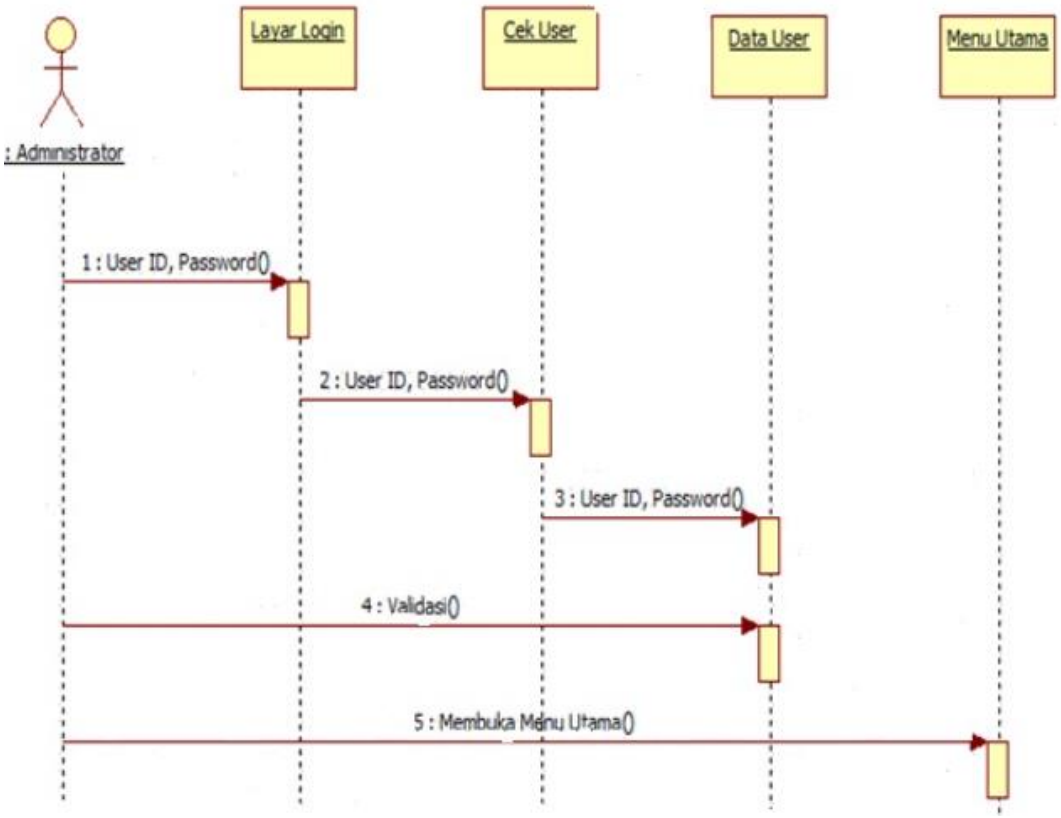
Дијаграм секвенце се користи за приказ скупа порука које су размењене између објеката који су у одређеној сарадњи како би се нека операција остварила. Овај дијаграм се приказује у две димензије а то су:

- Вертикала која показује време
- Хоризонтала која показује одређене објекте

Као што је већ речено, дијаграм служи за приказивање порука које су се проследиле између објеката у одређеном временском интервалу. Те везе, односно поруке које се прослеђују између објеката директно утичу на дужину линије живота сваког објекта. Линија живота није ништа друго него линија активности која показује колико је објекат био активан, односно колико дуго је објекат размењивао поруке са неким другим објектом. Овај дијаграм такође садржи и фокус управљања који служи за приказивање временског периода у којем је објекат био активан.

Елементи дијаграма секвенце се приказују на следећи начин:

- Објекти се приказују на раније описан начин
- Везе између објеката које су приказане хоризонталном линијом
- Линија живота објекта која представља испрекидану вертикалну линију
- Фокус управљања који се представља вертикално постављеним правоугаоником



Слика 7. Дијаграм секвенце

Сваки правоугаоник на врху дијаграма секвенце преставља неки објекат. Уколико се деси да неки објекат има Име1:Име2 то значи да тај објекат има два имена. Оквири на врху садрже име класе и име објекта. Усправне линије представљају животни век објекта. Хоризонталним линијама приказана је комуникација између објеката

7. Пример пројекта пружања услуга амбуланте

Задатак овог примера је да се оствари модернизација амбуланте уз помоћ информационих технологија, чиме ће се функционисање те амбуланте бити ефикасније као и једноставније.

У делу модернизације амбуланте мисли се на телефонско заказивање прегледа код жељеног лекара, чиме ће се смањити чекање пацијената испред радних просторија. Такође, овим постпуком се омогућава чување свих података о пацијентима у одговарајућој бази података где ће бити смештене информације о њиховим дијагнозама, заказаним прегледима и преписаним терапијама. Примарни кључ за претраживање свих ових података биће број картона пацијента.

Носиоци улога у овим дијаграмима су доктор, медицинска сестра и пацијент.

На слици 8 приказан је дијаграм случајева употребе у реалном окружењу. Као што је речено, дијаграм случајева употребе служи за остваривање комуникације, односно за грубо представљање функционалности неког система. Носиоци улога у овом случају су пацијент и запослени, односно доктор и сестра. Неопходно је да пацијент изврши уплату осигурања као прву интеракцију са системом. Уплатом осигурања долази се и до формирања картона. Након што је картон формиран, потребно да се закаже преглед код медицинске сестре. Када се преглед закаже, пацијент може доћи у заказаном термину на преглед. Преглед даље води до питања да ли је пацијенту потребно лечење или не. Уколико је пацијенту потребно лечење, доктор ће извршити ту интеракцију. Да би се извршило лечење, претходно доктор мора да провери стање картона пацијента, да провери да ли је можда пацијент алергичан на неке лекове који би били део терапије. Уколико је све у реду онда се започиње лечење при чему се извештај лечења води као ажурирање картона. Пре издавања рецепта пацијенту, неопходно је да медицинска сестра отвори рецепт.

На слици 9 приказан је дијаграм класе у реалном окружењу. У овом дијаграму описане су класе, а такође су одређени и типови атрибута који описују те класе. За пример класе чије је назив амбуланта објекти су: бројТелефона типа int, место типа string, адреса типа string, бројДоктора типа int, бројМедСестре типа int, бројПацијента типа int, радноВреме типа string. У класи амбуланта налази се листа за предлег пацијената и запослених у коју могу да се уносе нови пацијенти и запослени. Класа запослени описана је атрибутима име типа string, презиме типа string, датумЗапослења типа date, бројТелефона типа int, идентификациони број ИД типа int, радноВреме типа string и смена типа string. Класа доктор описана је атрибутима специјализација типа string, радноИскуство типа int, ИД доктора типа int као и листе које служе за унос термина прегледа пацијената. ПримиПацијента, погледајКартон и остало су активности које доктор обавља и које су

променљиве у зависности од пацијента до пацијента. Класа медицинскаСестра је описана атрибутом ИД која је типа int. Сви остали атрибути су улазне информације које се као у случају класе доктора, мењају у зависности од потребе. Отворикартон, узмиКартон, прегледајКартон и тако даље, то су све атрибути који се мењају у зависности пацијената уколико неки пацијент нема картон то значи да ће бити отворен картон, уколико има то значи да ће се картон узети из архиве и тако даље. Атрубит Термин описан је следећим атрибутима: датумТермина који је типа date, времеСати типа int, времеМинута типа int, доктор типа string, пацијент типа string. Ту се још могу додати атрибути ТерминПрегледа који се остварају уколико је термин заказан и који се уништавају уколико је термин одржан и завршен. Класа са називом Пацијент садржи атрибуте, односно информације о самом пацијенту. Ту се налазе атрибути као што су: име типа string, презиме типа string, имеРодитеља типа string, јмбг типа int, број картона типа int и тако даље. Као и у претходним случајевима то су атрибути који представљају излазне информације (као што су ови наведени) и атрибути који представљају улазне информације (дајПодатке, дајБројКартона, закажиПреглед, уплатиОсигурање,...). Последња класа у дијаграму класа јесте класа преглед која је описана одређеним атрибутима и такође садржи атрибуте који се по потреби креирају или уништавају, као што је било у случају класе Доктор.

На слици 10 приказан је дијаграм објекта за случај амбуланте. Дијаграм Објекта служи за детаљнији опис класа. За разлику од дијаграма класа, објекти у дијаграму објеката уместо типа, који је већ декларисан у дијаграму класа, имају опис. Класа студентскаАмбуланта : Амбуланта је описана атрибутима као што су: бројТелефона = 034336773, место = „Крагујевац“, бројДоктора = 1, бројМедСестара = 2 и тако даље. Код дијаграма Објеката, за разлику од дијаграма класа, не постоје улазни атрибути нити они који су привремени као што је тамо био терминПрегледа. На истом принципу су урађењне у све друге класе у дијаграму објекта.

На слици 11 приказан је дијаграм активности у којем је описан принцип отварања картона у амбуланти. Дијаграм активности се дели на два дела. Један део чини пацијент док други део чини медицинска сестра. Пацијент је у дужности да пре свега дође у амбуланту. Доласком у амбуланту, медицинска сестра је у обавези да провери његов картон.Уколико је пацијент има картон, онда ће се извршити провера истог. Уколико пацијент нема картон потребно је да поднесе захтев за отварање картона. Подношењем захтева за отварање картона, потребно је да да личне податке. Након датих података у бази се проверава да ли је пацијент осигуран. Уколико је пацијент осигуран може да отвори картон. Уколико пацијент није осигран потребно је да уплати осигурање и да потврду о уплаћеном осигурању донесе у амбуланту. Уколико потврду не донесе у року од 72 сата, постаће неважећа и регистрација ће бити одбијена. Када је донео потврду о уплаћеном осигурању, медицинска сестра одобрава регистрацију и шаље обавештење да је картон отворен.

На слици 12 приказан је дијаграм активности у којем је описан принцип заказивања прегледа у амбуланти. Учесници у овом дијаграму су пацијент, медицинска сестра и доктор. Као први корак, пацијент мора да преда здравствену књижицу медицинској сестри. Медицинска сестра проверава базу пацијената. Уколико пацијент нема картон онда се врши отварање новог картона, а уколико има картон онда се разматра следећи проблем а то је термин. Уколико пацијент има заказан термин, онда се шаље директно код доктора на преглед чиме се завршава интеракција. Уколико пацијент нема заказан термин онда се то може извести на два начина. Први начин заказивања термина је долазак у ред, односно чекање у реду, док је други начин да се термин закаже неким другим путем, или преко телефона или преко интернет странице. Када се термин закаже, онда пацијент може доћи на преглед.

На слици 13 приказан је дијаграм активности у којем је описан преглед пацијента у амбуланти. Учесници овог дијаграма су пацијент, медицинска сестра и доктор. Доласком у амбуланту, пацијент мора да уђе у ординацију. Уласком у ординацију, пацијент даје број картона сестри на основу чега она тражи картон пацијента у бази картона. Пронађени картон пацијента прослеђује се доктору. На основу картона, доктор прима пацијента. Потребно је извршити стање пацијента и видети да ли пацијент болује од акутне или хроничне болести. Уколико је реч о хроничној болести, потребно је извршити преглед пацијента или променити терапију. Уколико је реч о акутној болести онда треба проценити стање пацијента. У ова случаја, доктор треба да одлучи да ли је потребно издати рецепт за лек или лек није потребан. Уколико је лек потребан, онда је потребно издати и рецепт за лек. Који год да је случај у питању, након прегледа, доктор уписује резултате у акртон пацијента. Картон прослеђује медицинској сестри где она даље оверава картон.

На слици 14 приказан је дијаграм секвенце за случај заказивања прегледа у амбуланти. Учесници дијаграма су пацијент, медицинска сестра и база података. У првом делу под називом „тражи доктора“ види се интеракција између пацијента и базе података. Пацијент шаље користи базу података како би пронашао термин када је неки доктор слободан. Као повратну информацију, пацијент добија директно из базе који доктор је слободан у датом термину. Уколико пацијенту не одговара доктор или термин он може одустати од прегледа, као што је учињено на дијаграму. Уколико се пацијент ипак одлучи за преглед неопходно је да своје податке проследи медицинској сестри одакле ће она те податке проследити бази података како би могла да пронађе картон пацијента. Уколико се пацијент налази у бази података, медицинска сестра ће добити повратне информације из базе. У другом делу под називом „претрага картона“ пацијент комуницира са медицинском сестром и заказује преглед. Медицинска сестра узима картон из базе. Након узимања картона, медицинска сестра комуницира са новом класом а то је ТерминПрегледа, где је потребно евидентира термин прегледа пацијента. Повратну

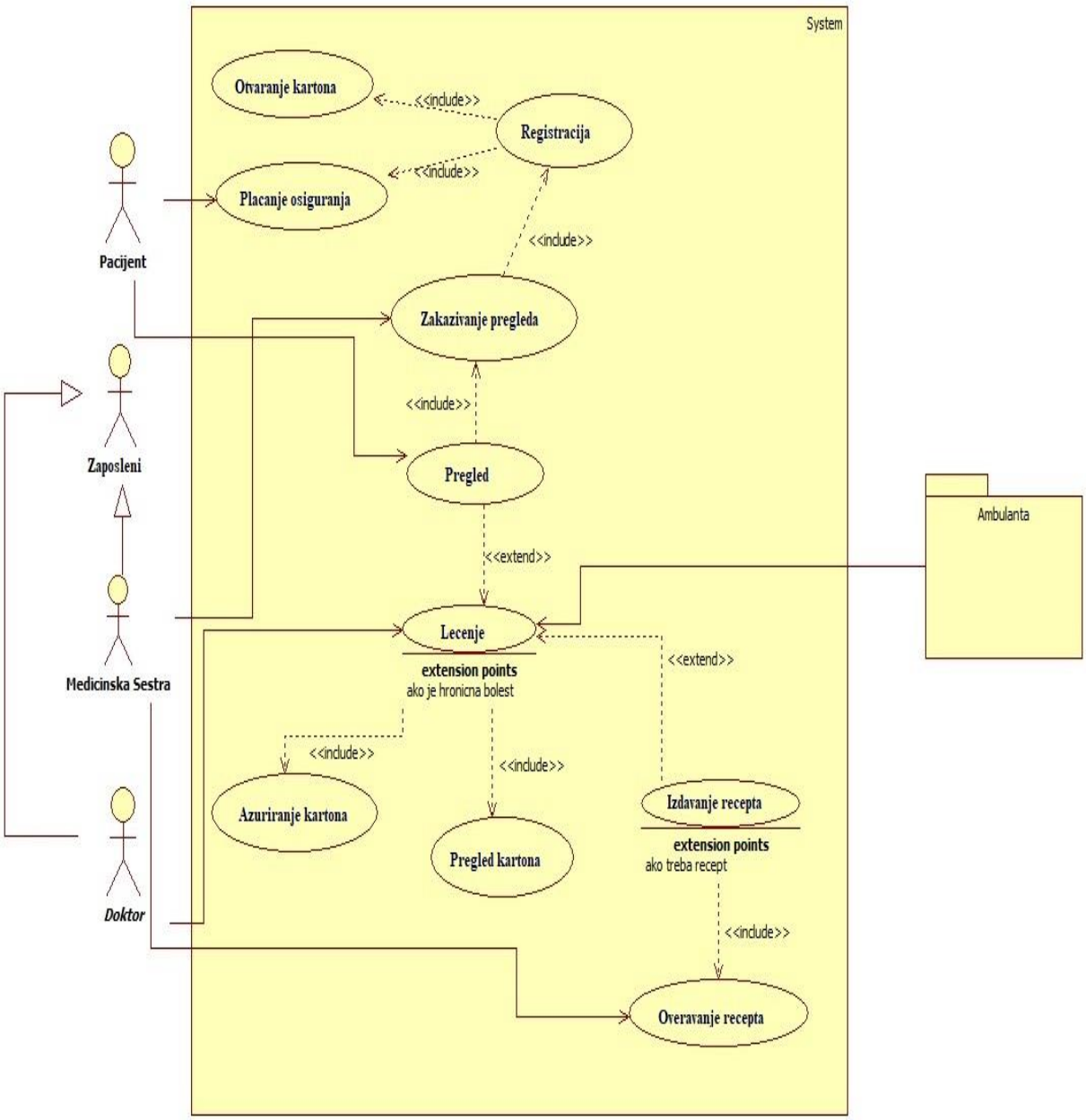
информацију о заказаном термину добија медицинска сестра након чега она ту информацију прослеђује пацијенту. У случају да пацијент нема картон биће обавештен од стране медицинске сестре. Да би се извадио нови картон, пацијент мора да проследи своје податке медицинској сестри након чега ће она те податке проследити новој класи под именом МедицинскиКартон. Након тога поставља се питање да ли је пацијент осигуран? Уколико пацијент нема осигурање потребно је да га уплати. Уколико пацијент има осигурање потребно је да потврду о уплати осигурања проследи медицинској сестри. Након добијене потврде о уплати, медицинска сестра ће јавити пацијенту да је његов картон отворен, у супотроном, биће обрисан захтев за отварање картона као и захтев за термин прегледа, услед чега ће пацијент као одговор добити да су његови захтеви за отварање картона као и захтев за преглед бити одбијени.

На слици 15 приказан је дијаграм секвенце за случај прегледа у амбуланти. У овом случају главне класе које комуницирају су пацијент, медицинска сестра, доктор и база података. Потребно је да пацијент да број картона медицинској сестри. На основу броја картона, она прослеђује те податке бази података одакле добија податке о пацијенту, односно добија картон. Након добијеног картона, медицинска сестра прослеђује картон доктору. Након узимања картона, доктор прима пацијента у ординацију. Након уласка пацијента у ординацију, доктор прегледа картон. Уколико је рец о хроничној болести, пре прегледа доктор мора да прегледа историју болести које је пацијент имао, а након тога може да прегледа пацијента. Након прегледа, доктор добија резултате које ће уписати у картон. Уколико није реч о хроничној болести, доктор ће прегледати пацијента на основу чега ће знати стање његовог здравља. У ова случаја, потребно је да доктор евидентира преглед а то ће урадити у новој класи „Преглед“. Уколико је потребан рецепт, доктор ће то такође евидентирати у класи „Рецепт“ и након тога ће пацијенту преписати рецепт. Пацијент рецепт предаје медицинској сестри након чега она врши оверу рецепта и то бележи у класи „Рецепт“. Након архивирања оверења рецепта, рецепт се прослеђује пацијенту. Након одласка пацијента, сестра ажурира картон и прослеђује га бази података. База података као информацију враћа ажурирани картон након чега медицинска сестра ажурирани картон смешта назад у базу података.

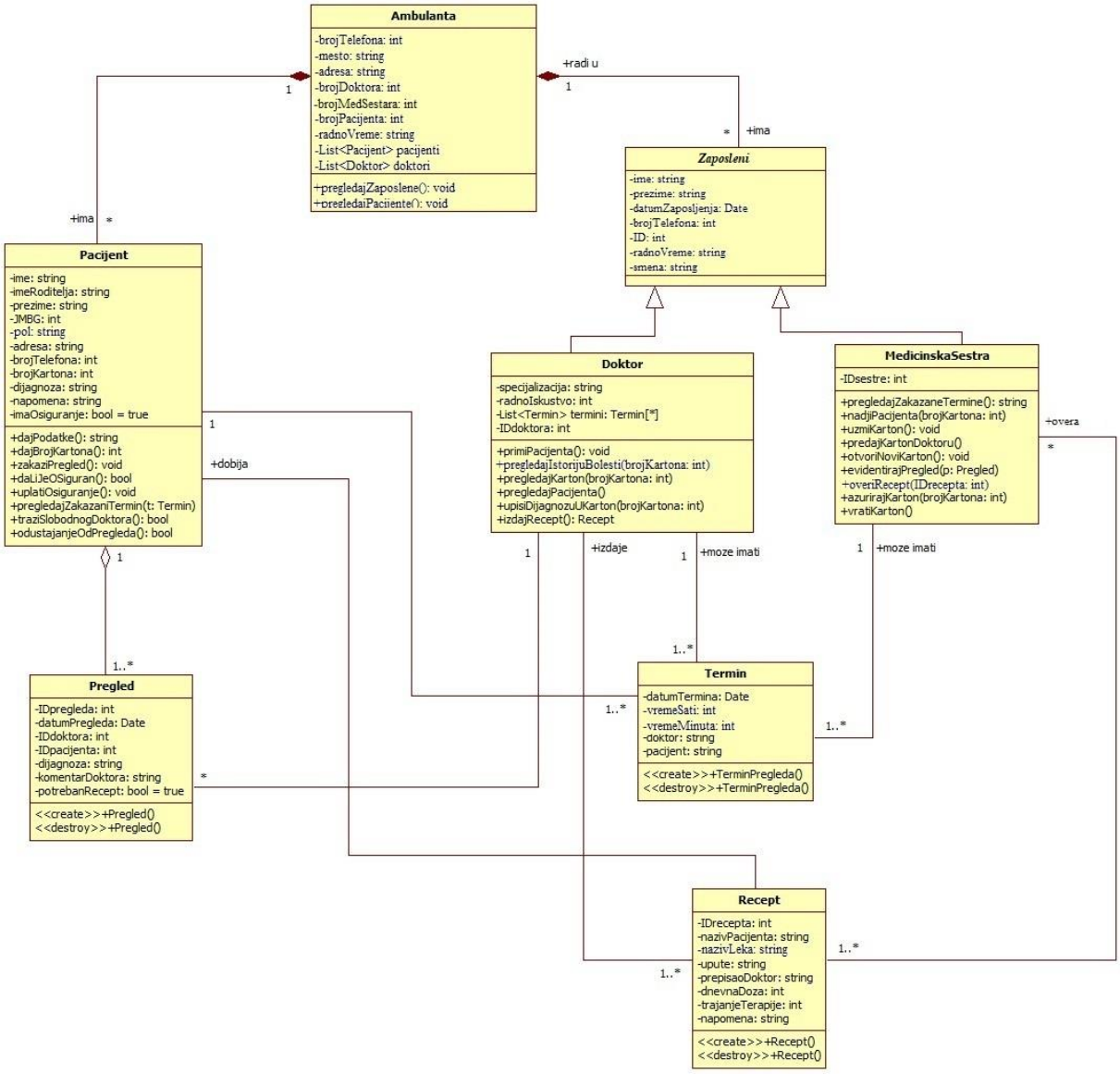
На слици 16 приказан је дијаграм компоненти за случај амбуланте. На овом дијаграму јасно се може видети да када се корисник пријави да може да изабере једну од понуђених опција. Пријавом, бира се да ли је запослени медицинска сестра или доктор. Класа МедицинскаСестра може да комуницира са осталим компонентама. Уколико је потребно да медицинска сестра отвори картон пацијенту, комуницираће са пакетом „Отварање картона“ у којем се налазе класе МедицинскаСестра и Пацијент, јер су те две класе потребне за отварање картона пацијенту. Такође, медицинска сестра може да комуницира и са осталим пакетима, а то су: регистрација, наплаћивање осигурања и заказивање прегледа. Са друге стране, након пријављивања, доктор може да комуницира са пакетима као што су: преглед стања, ажурирање новог стања и издавање рецепта.

На слици 17 приказан је дијаграм пакета за случај амбуланте. Дијаграм пакета служи за груписање сродних елемената, а најчешће се користи за груписање класа. У овом случају постоје пакети у пакету. Први пакет јесте Амбуланта у који су смештена друга два пакета а то су МедицинскиКартон и Особље. У пакету МедицинскиКартон смештене су класе: пацијент, преглед, термин и рецепт. У пакету Особље смештена је класа Запослени у коју спадају класе Доктор и МедицинскаСестра. Види се да пакет МедицинскиКартон податке добија од пакета Особље. Пакет Амбуланта добија информације из пакета Апликација. У пакету Апликација налазе се два друга пакета, КорисничкеФорме и КорисничкеКонтроле. У пакету КорисничкеФорме спадају класе: доктор, медицинска сестра и „main app“. У пакет КорисничкеКонтроле спадају класе: уносПацијента, прегледПацијента, прегледајПацијента, прегледПацијента, испишиРецепт, закажиПреглед и прикажиТермине.

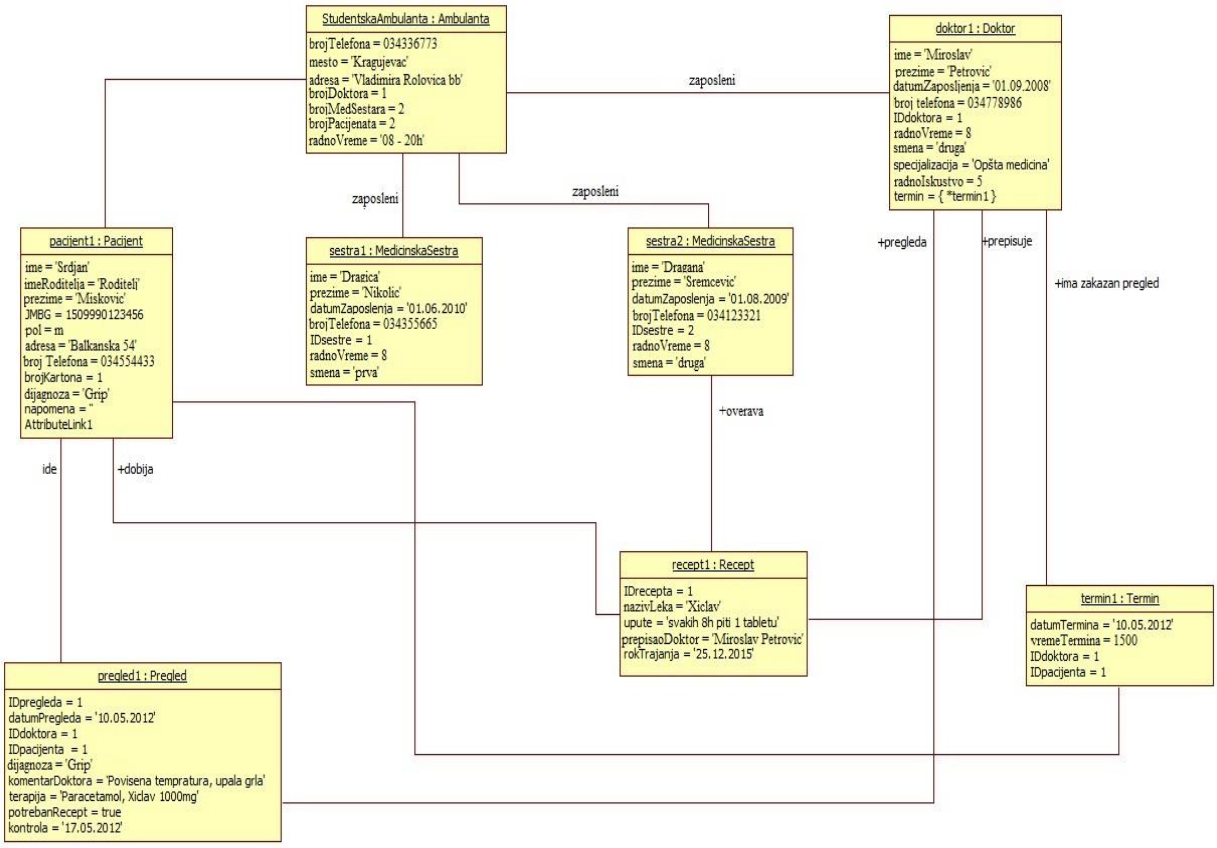
На слици 18 приказан је дијаграм распоређивања за случај амбуланте. Намена ових дијаграма јесте да прикажу хардверску архитектуру система, односно приказују распоређивање делова система на физичким локацијама.



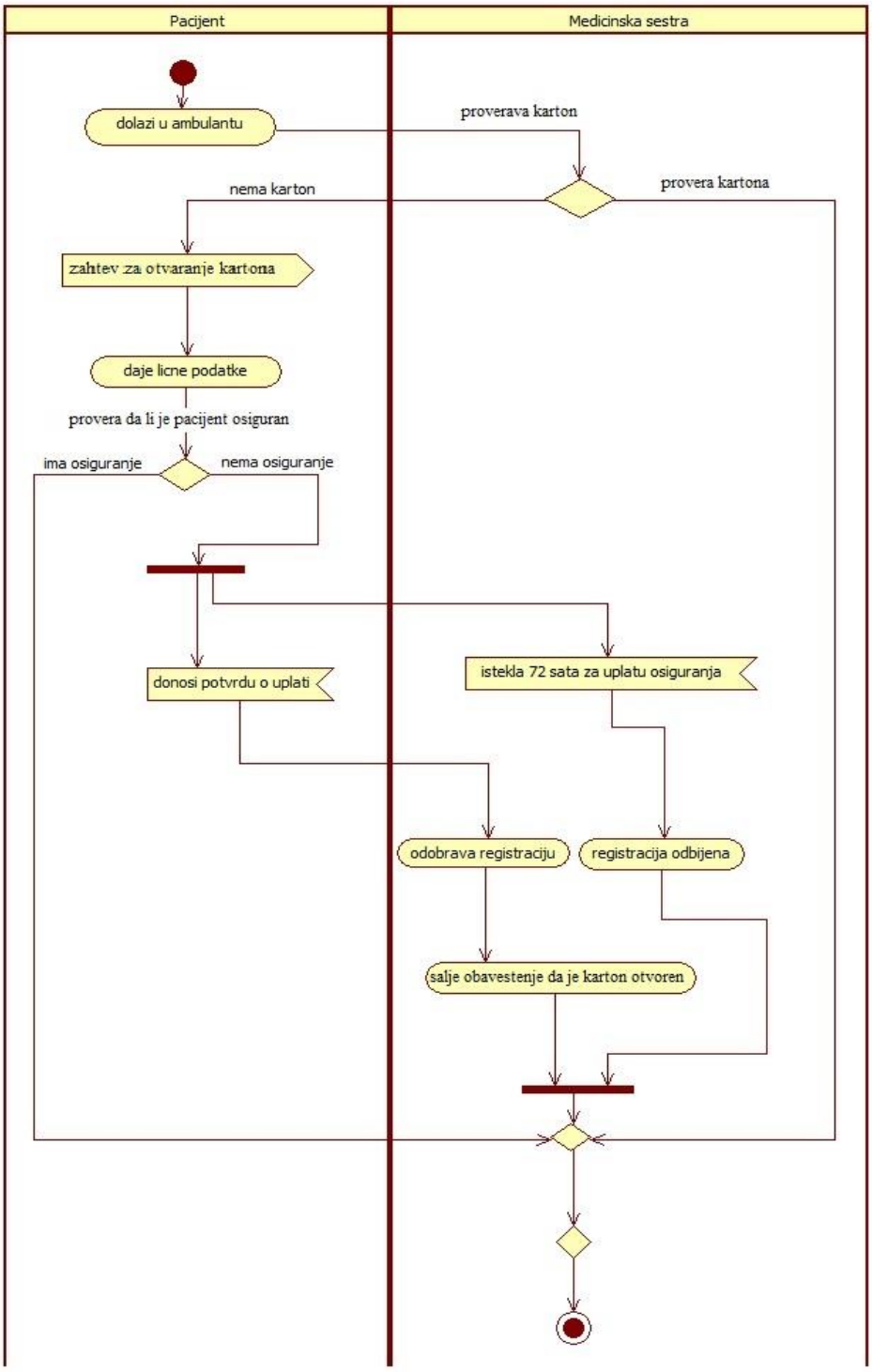
Слика 8. Дијаграм случајева употребе у амбуланти



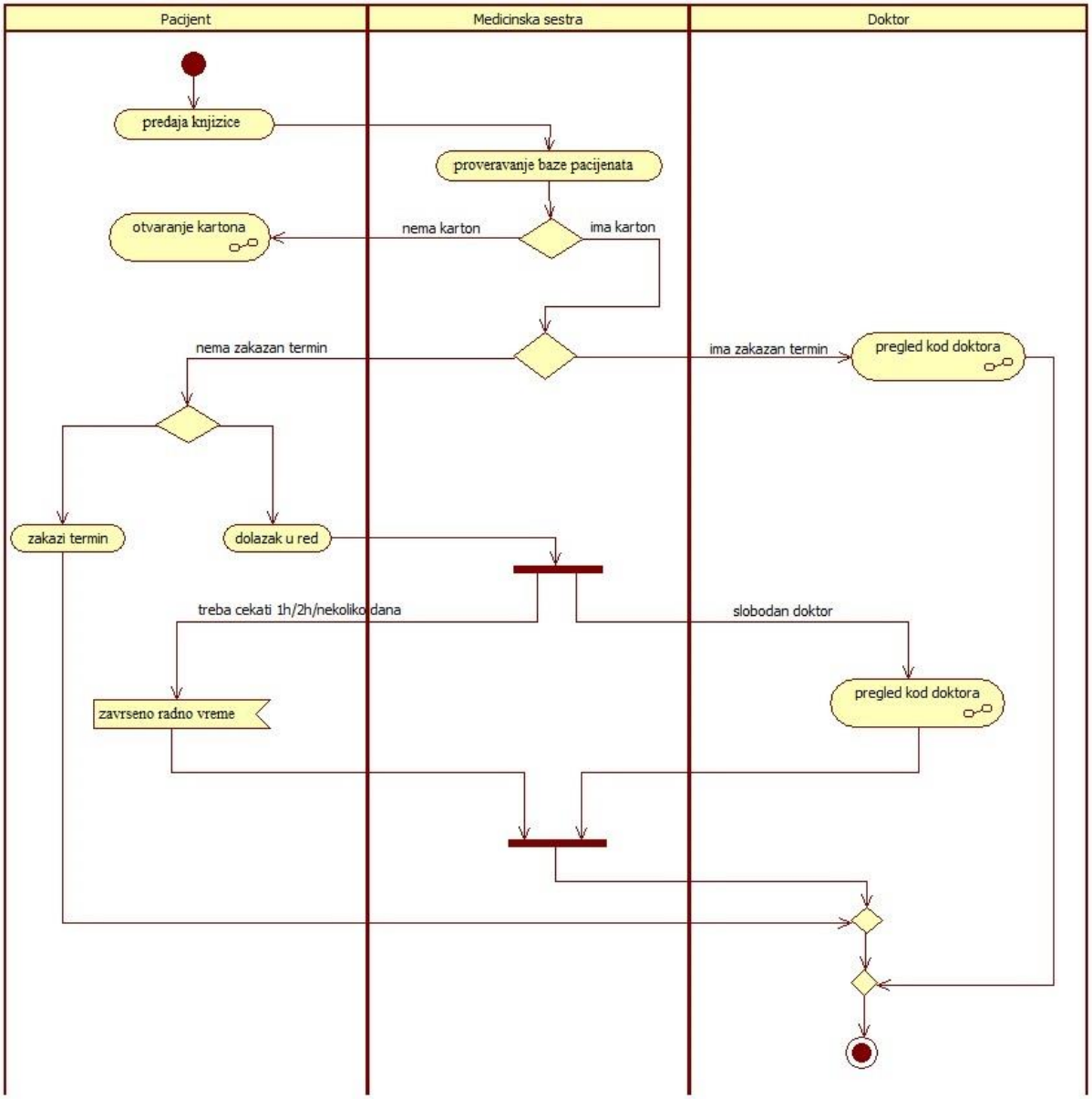
Слика 9. Дијаграм класе за случај амбуланте



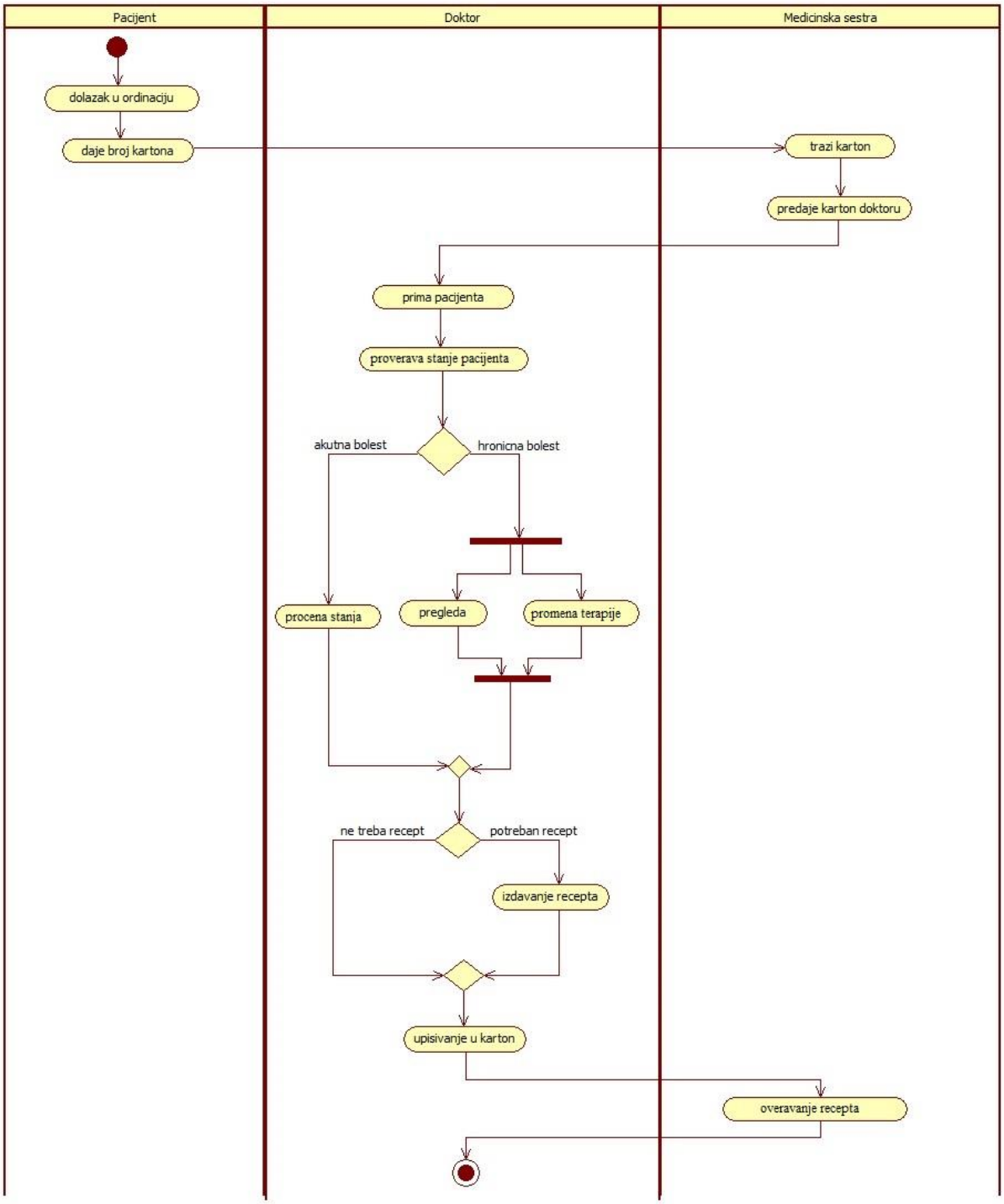
Слика 10. Дијаграм објеката за случај амбуланте



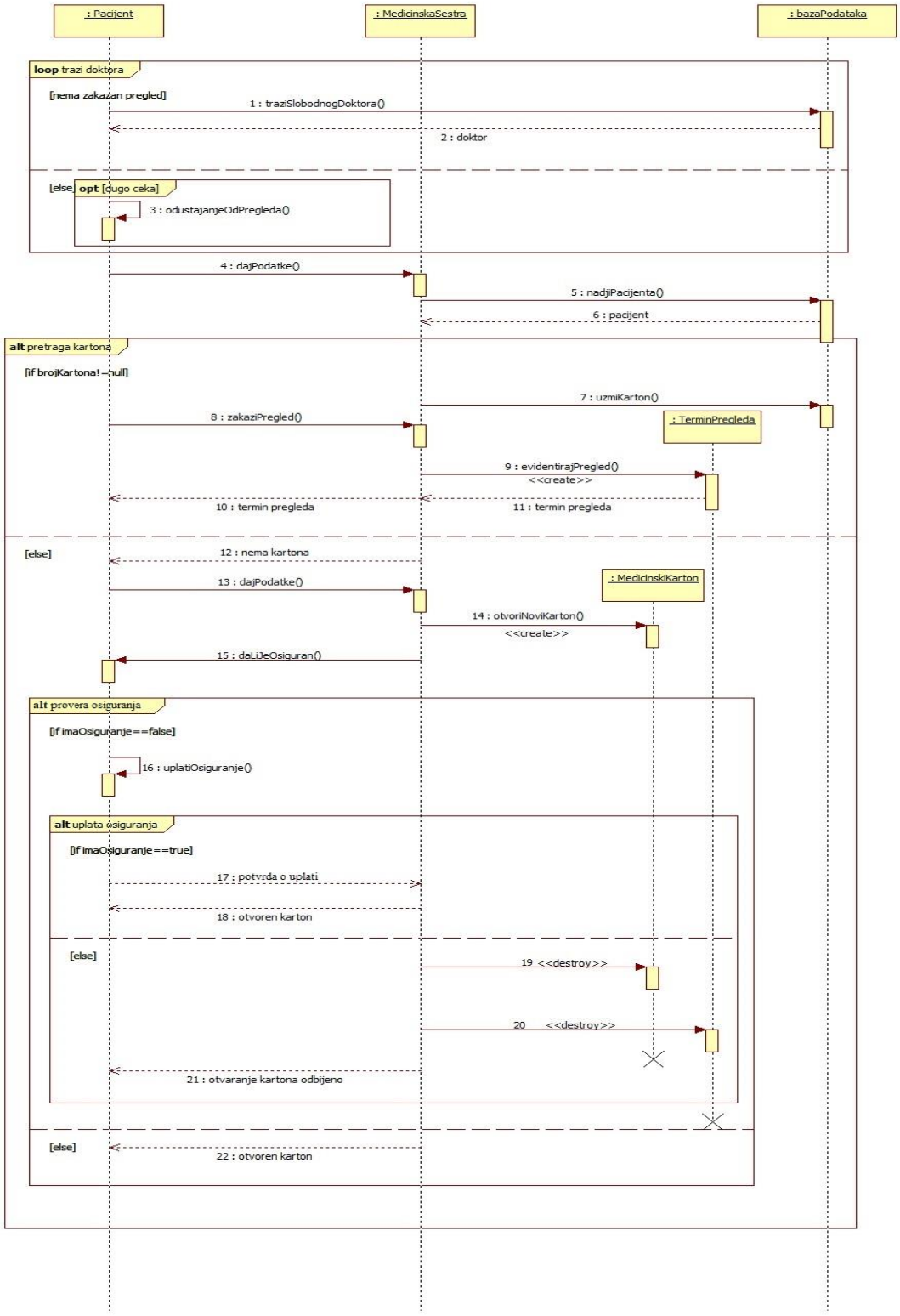
Слика 11. Дијаграм активности (отварање картона) за случај амбуланте



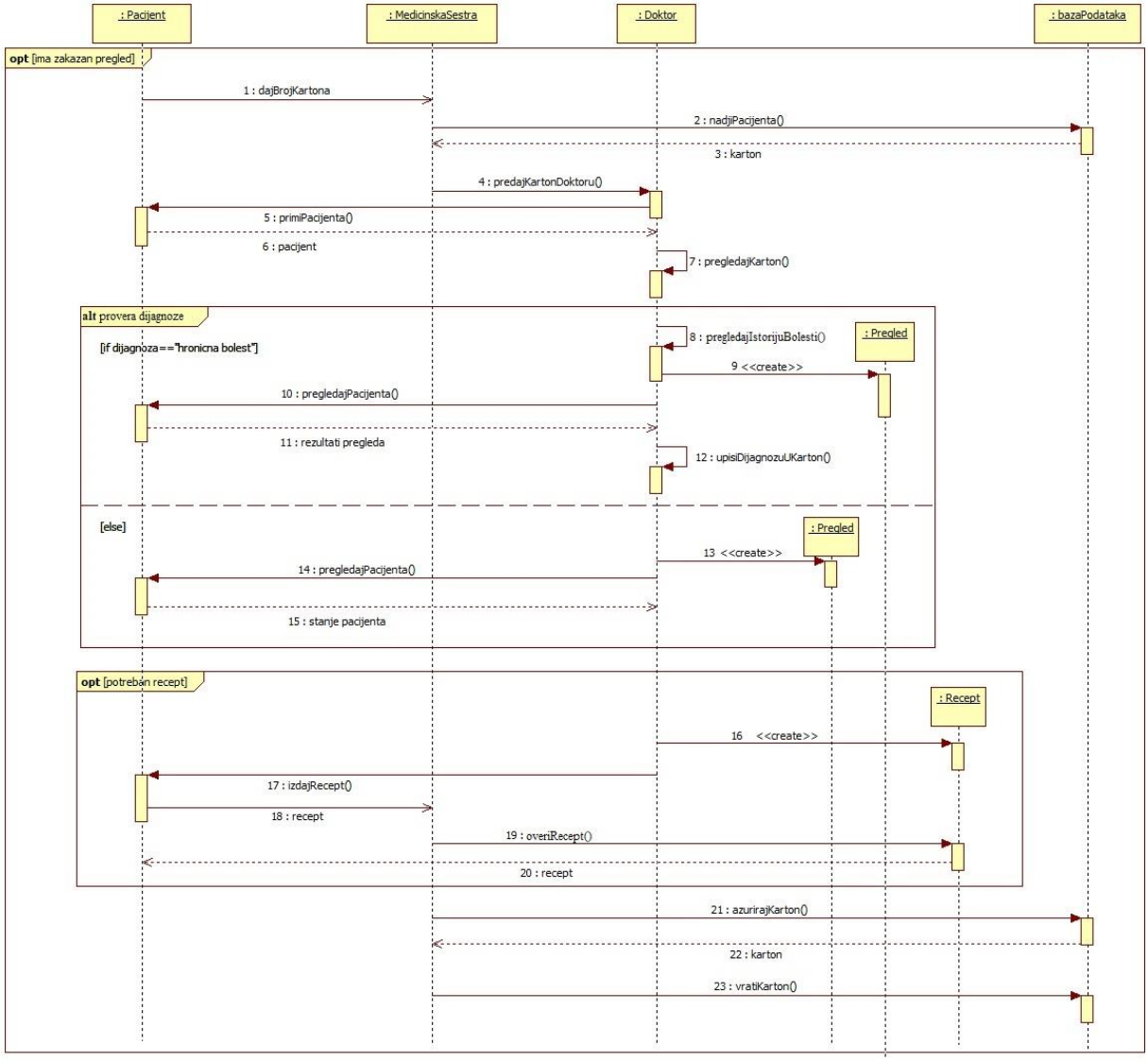
Слика 12. Дијаграм активности (заказивање прегледа) за случај амбуланте



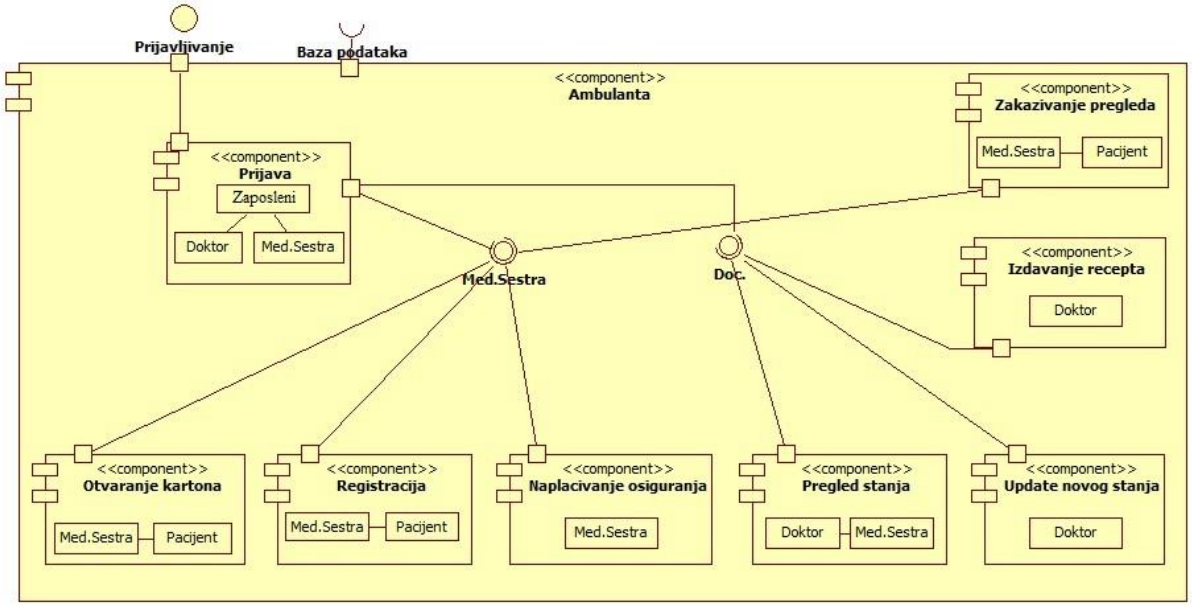
Слика 13. Дијаграм активности (преглед) за случај амбуланте



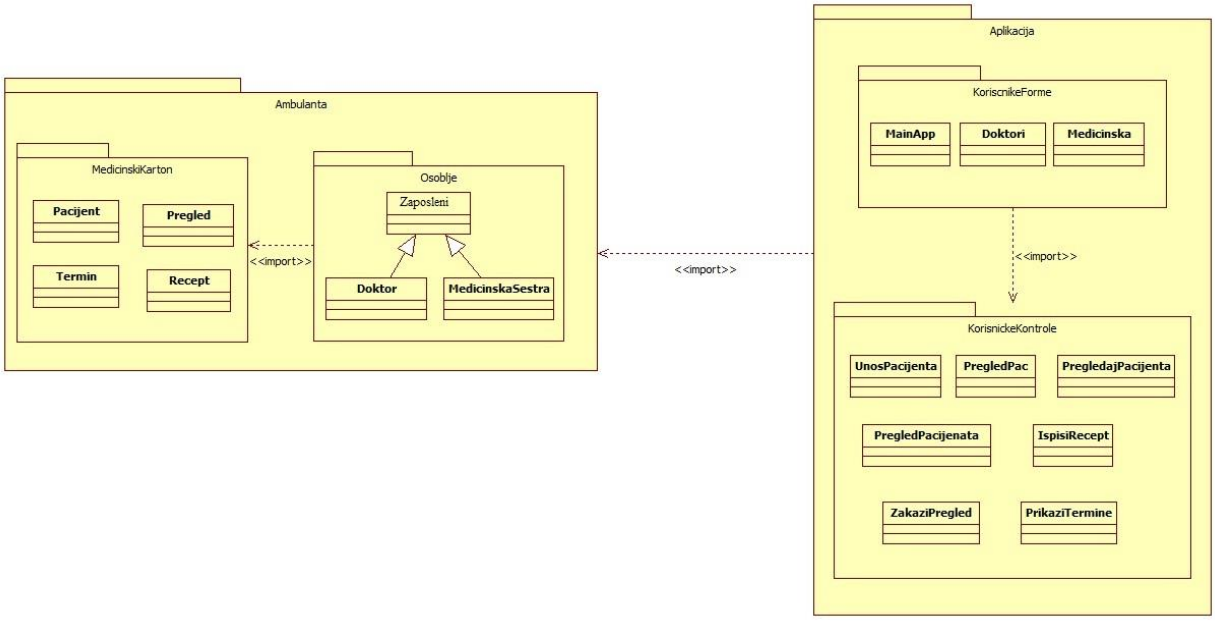
Слика 14. Дијаграм секвенце (заказивање прегледа) за случај амбуланте



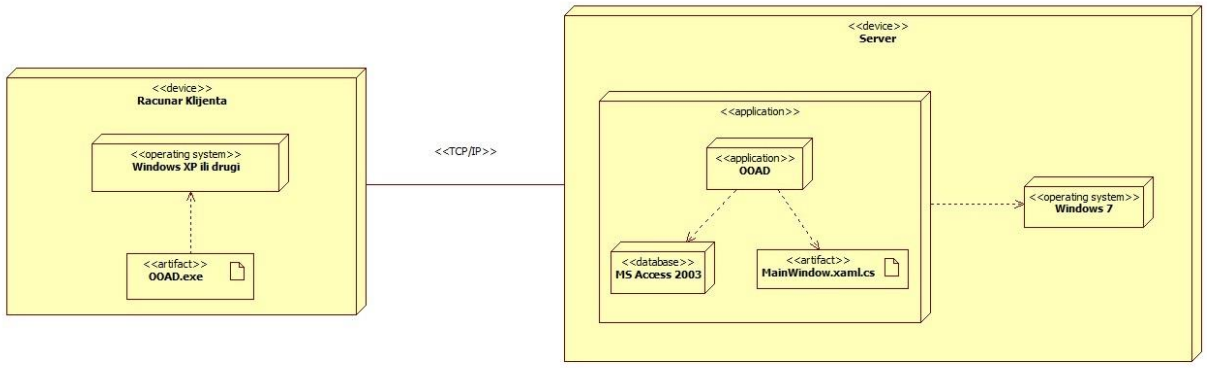
Слика 15. Дијаграм секвенце (преглед) за случај амбуланте



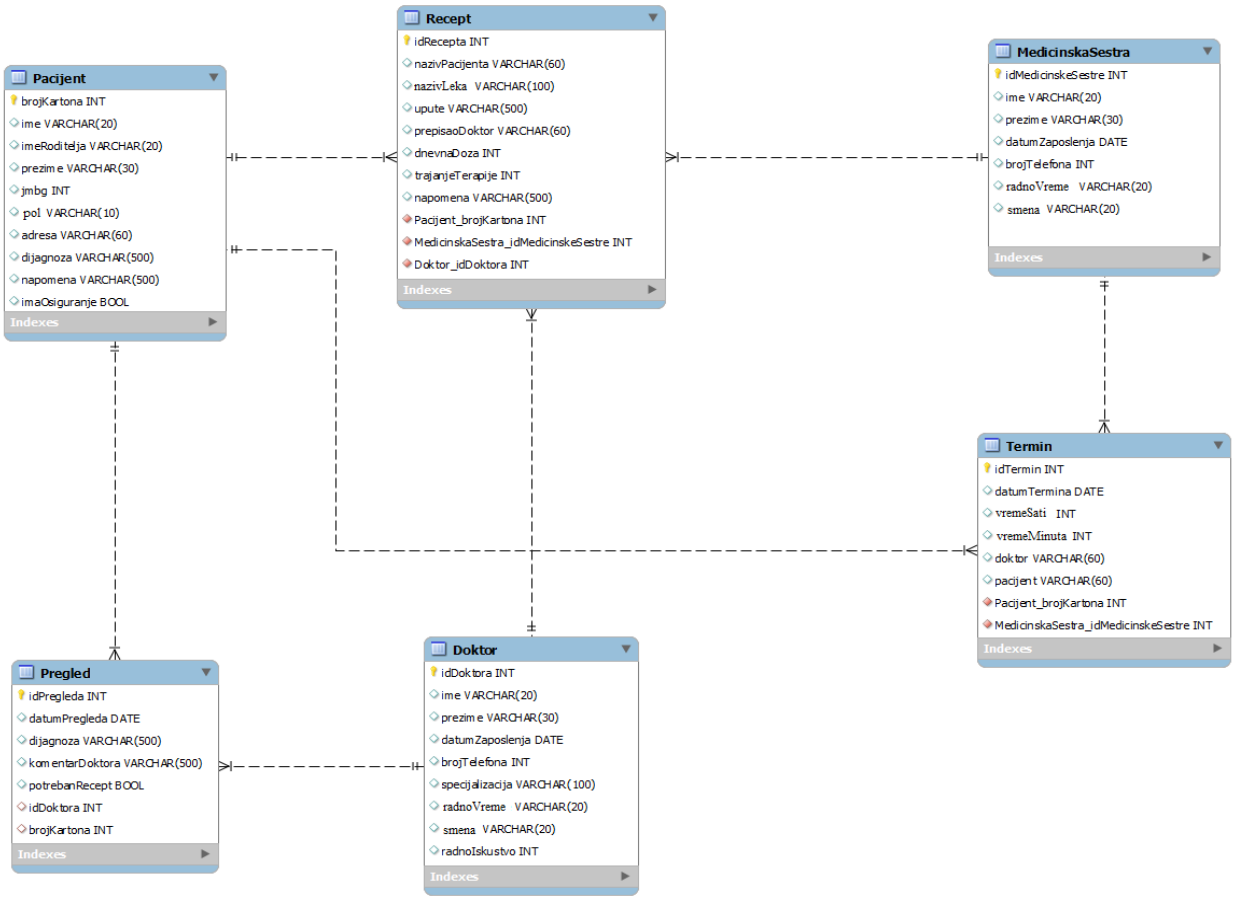
Слика 16. Дијаграм компоненти за случај амбуланте



Слика 17. Дијаграм пакета за случај амбуланте



Слика 18. Дијаграм распоређивања за случај амбуланте



Слика 19. Концептуални модел база података

8. Закључак

У овом раду дат је комплетан опис UML језика за моделирање који је захваљујући својој једноставности, а у исто време и својој способности да детаљно прикаже функционалност неког система постао стандард. У уводном делу било је речи о томе шта је уопште UML и која је намена самог језика. Дат је кратак историјат о самом језику и због чега и како је он настао. Описани су сви дијаграми, они најбитнији па и они мање битни, али у сваком случају битни, јер да нису, сами творци UML-а их не би држали као део језика. Поред тога што је дат опис свих дијаграма којим UML располаже, дати су и графички примери сваког дијаграма понаособ. Такође, дат је и пример UML дијаграма у неком реалном систему, који је у овом случају био државна установа, односно амбуланта. Дијаграмима је објашњена комуникација између особља установе и пацијената, као самих корисника. Све интеракције између корисника и особља приказане су у одговарајућим дијаграмима, један од пример тих дијаграма јесте дијаграм случајева употребе, као и унутрашња комуникација објеката која је дата у секвенцијалним дијаграмима.

Коришћењем UML дијаграма поједностављује се рад неког система, једноставније се увиђају недостаци што проузрукује лакше решавање самих проблема, односно отклањање недостатака. Свакако би требало наставити са усавршавањем UML-а јер што више могућности сам језик има, лакше је представити неко реално окружење са свим предностима и недостацима који се у њему налазе.

Литература

[1]. Вељовић А.: **Основе објектног моделирања - UML**, Компјутер библиотека, 2012.

[2]. Naiburg E., Maksimchuk R.: **UML for Database Design**, Addison-Wesley, Boston, 2012.

[3]. Grady Booch, James Rumbaugh, Ivar Jackobson: UML водич за кориснике, 2000.

[4]. James Rumbaugh, Ivar Jackobson, Grady Booch: The Unified Modeling Language Reference Manual 2nd edition, Addison-Wesley Professional, Јун 29, 2004.