Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Мастер академске студије

Модул: Индустријски инжењеринг

Предмет: Пројектовање информационих система и база података

Број индекса: 397/2013

Бојана М. Лазаревић

Развој база података у UML-у

мастер рад

Комисија за преглед и одбрану:

1. **Др Милан Ерић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру мастер рада са темом "*Развој база података у UML-у*" кандидат треба да:

Кроз теоријске основе визуелног пројектовања база података као полазишта и конкретизацију кроз практичну студију прикаже примену UML-а у развоју база података. Рад треба да обухвати уводна теоријска разматрања везана за развој база података, елемнте односно концепте UML-а везане за пројектовање база података, примену истих на конкретном реалном систему. На крају дати закључна разматрања.

Препоручена литература:

1. Blaha M., Premerlani W.: **Object-Oriented Modeling and Design for Database Applications**, Prentice'Hall, Inc., New Jersey, 1998.
2. Вељовић А.: **Основе објектног моделирања – UML**, Компјутер библиотека, Чачак, 2002.
3. Naiburg E., Maksimchuk R.: **UML for Database Design**, Addison-Wesley, Boston, 2012.
4. Лазаревић Б.: **Базе података**, ФОН Београд, Београд 2003.
5. Полишчук Ј.: **Пројектовање информационих система**, Електротехнички факултет, Подгорица,2007.
6. Rainer K., Turban E.: **Introduction to Information Systems**, John Wiley & Sons, Inc., 2009

Изјава о самосталној изради рада

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

397/2013 Бојана Лазаревић

_____

(потпис)

Садржај

1.0. УВОДНА РАЗМАТРАЊА

1.1. Предмет истраживања

Данашње време и окружење у ком живимо диктира другачији темпо живота који је доста променљив и непредвидив. Те чињенице са собом повлаче и динамичност у пословном свету, па самим тим и на тим пољима долази до наглих промена, па је веома тешко испратити све промене и благовремено реаговати на исте.

Савремени пословни свет би био незамислив уколико се не би користиле савремене базе података као и савремени алати за пројектовање и развој истих. Имајући у виду да базе података представљају скуп односно колекцију података који треба да буду добро организовани, које ће омогућити брзо претраживање и приступ, потребно је и одабрати прави алат и методу за пројектовање базе података.

Из ових захтева је настао овај рад чији је предмет анализа пројектовања база података у UML-у, као и основа функционисања и примене као и конкретне употребе истог односно моделирање са студијском анализом.

Полазиште рада представљају бројни научни радовии електронски извори као и препоручена литература и остала доступна литература у електронском и писаном издању.

1.2. Значај и актуелност истраживања

Значај овог истраживања се огледа пре свега на општост UML-а, као и чињеницу која се односи на неусаглашеност ранијих програмских језика који су се користили за визуелизацију неких пословних система, па је и сам UML настао као обједињење више метода објектно оријентисаних графичких језика који су се користили за пројектовање. Деведесетих година је постало јасно да се мора наћи неки споразум везан за саму технологију као и технике како би се објектно-оријентисане идеје представиле. Након тога је следило више покушаја који нису успели, да би крајњи производ био нови метод. Тек 94-те године долази успех и тада настаје први предлог развојне методе, да би годину дана касније организација Object Management Group тражила да се напише предлог за

стандардизацију приступа развоју са објектно-оријентисаним методологијама. 1997. године UML бива прихваћен за неку врсту стандарда, и први UML стандард је постао званичан у новембру 1997. године. [1]

Значај и примена UML-а је изузетно велика, што сведочи велики број књига, часописа, присуство у медијима и путем интернета. Постоји и велики број софтвера који се базирају на овом стандарду, што ће бити објашњено даље у раду.

1.3. Циљеви истраживања

Генерално посматрано, циљ сваког истраживања јесте сазнање и приступ неком проблему као и предлог решења истог, који је дат предметом истраживања. У истраживању је потребно доћи до сазнања нових појава, својстава као и суштине проблема и праксе у целини.

Циљ овог истраживања односно рада јесте да се детаљно објасне сви појмови везани за визуелно моделирање. То подразумева и функционисање и примену UML-а, као јединственог програмског језика који се користи за визуелизацију, спецификацију, конструцију и документовање софтверких решења.

1.4. Хипотезе истраживања

Сваки научно-истраживачки рад поседује елементе у којој се налази мисао самог проблема и истраживања, међу којима се налази и хипотеза. „Хипотеза је одраз теоријског и искуственог мишљења истраживача или методолога који је поставио проблем, циљеве и очекивано решење“.

У складу са претходном реченицом, хипотеза научно-истраживачког рада гласи:

UML (Unified Modeling Language)je обједињени визуелни језик, пројектован као врло флексибилан и прилагодљив, за пословно и софтверско моделовање у свим фазама развоја и за све типове система, као и за генерално моделовање којим се дефинишу статичке структуре и динамичко понашање.

Додатне хипотезе су:

- Циљ визелног моделовања јесте комуникација

- Три основна градивна блока UML-а јесу ствари, релације и дијаграми.

1.5. Методе истраживања

Приликом истраживања коришћене су методе анализе и синтезе теоријских основа и праксе као и промишљања и доказивања.

Истраживање у овом раду се базира на основу познатих основних метода управљања и моделовања, посредством одговарајућих модела за описивање одабраних проблема.

У циљу постављања јасног истраживачког задатка и његове успешне реализације, у раду су примењени емпиријски, критички и дескриптивни метод анализе.

1.6. Структура рада

Када је реч о структури самог рада, смештен је у шест делова, при чему је на врло јасан и прецизан начин раздвојен сваки део од наредног, док заједно чине једну комплексну целину.

Први део рада говори уопштено предмет и полазиште истраживања, укључујући и хипотезе као и методе истраживања, где су наведене оне које су коришћене у самом научно-истраживачком раду.

Други део рада говори о елементима UML-а, где се посебни акценат поставља на UML дијаграме (дијаграми класа, случајева употребе, секвенци и друго).

Трећи део рада обухвата уопштено моделирање база података у UML-у, као и основне елементе за дизајнирање истих. Такође је описан софтвер који је обрађен у даљем тексту односно Мицрософт Висио 2016.

Четврти део не мање важан од осталих јесте приказ конкретног проблема и његово решење коришћењем UML-а. За пример је узет информациони систем приватне болнице.

Пети део обухвата закључна разматрања, док шести део носи списак коришћене литературе.

2.0. ОСНОВЕ UML-а

UML представља обједињени језик за моделирање који се користи за спецификацију, визуелизацију, конструцију и представљање односно документовање детаља једног софтверског система. UML у себи садржи све информације о могућим одлукама али и пружа разумевање о датом систему који је у изради. Као што је већ поменуто, UML је настао као унија свих претходних искустава везаних за моделовање софтверких система па је самим тим као такав и постао стандард. UML је дизајниран тако да може да ради са интерактивним алатима који имају могућност да генеришу код документационих извештаја и програмски код аутоматски.

Имајући у виду да данашњи пројекти који су покривени софтвером садрже велики број линија кода, постало је готово немогуће разматрати пројекте у целини, и ући у суштину до најситнијих детаља, па се управо у томе и огледа велика предност UML-а, јер пружа организационе конструкте који могу да држе ову сложеност под контролом.

Пошто су поменути стандарди, битно је знати да су они неопходни да би се постигао циљ визуелног моделовања, а то је комуникација.

Комуникација између следећих категорија корисника, и то:

- Систем аналитичари и крајњи корисници
- Архитекте система
- Развојни инжењери
- Контролори квалитета,
- Руководиоци пројекта који воде и усмеравају кадрове и ресурсе. [2]

Комуникација се може остварити коришћењем невизуелних (текстуалних) информација, међутим људи визелно могу доста лакше и прихватити и уочити поједине детаље и ствари.

UML представља стандард који је усвојен од већине корисника и група за стандардизацију као што су ANSI и OMG (Object Managment Group).

2.1. Појам UML-а

UML данас спада у значајније језике и термин означава *Unified Modeling Language,* што у преводу значи систем за моделовање, односно пројектовање. [3]

При развоју софтвера и пројектовању базе података, велики проблеми настају већ у старту креирања решења, а то су непрецизни захтеви. UML нуди алате који служе управо за боље прикупљање захтева.

UML се користи за боље разумевање, дизајнирање, претрагу, конфигурисање, одржавање и контролу информација код сложених система. Он укључује шематичке концепте, нотацију и смернице, док у себи садржи статичке, динамичне и организационе делове. Битно је напоменути да UML не дефинише стандардни процес, али је намењен да буде од користи уз итерактиван процес развоја и да подржи већину постојећих објектно оријентисаних развојних процеса.

UML омогућава:

- **Визуелизацију**: визуелни, графички језик
- **Спецификацију**: формирају се прецизни, недвосмислени и потпуни модели
- **Конструкцију**: није визуелни програмски језик, али његови модели могу да се повежу непосредно са разним програмским језицима, али је и могуће пресликавање из UML-а у програмски језик
- **Документовање**: могу се документовати захтеви, архитектура, изворни код и друго . [4]

Разматрано на мало простији начин, вредност UML-а се може сагледати ако је потребно описати неки код не употребљавајући језик за моделовање (као што је UML), његово разматрање би било веома дугачко, непрецизно и збуњујуће, а самим тим UML “приморава” програмера да боље сагледа свој приступ.

2.2. Историја и развој UML-а

Сам развој UML-а се прати још од почетка 70-их година двадесетог века, када се заправо од тада па наредних 10 година и показала потреба за употребом објектно-оријентисаних језика.

Развој UML-а су подстакли и други језици, као што су Entity Relationship modeling, SDL (Specification and Description Language) и други. У периоду до 1994. године број објектно-оријентисаних језика се повећао на преко 50.

1994. године су Grady Booch и Jim Rumaugh (чланови Rational Software Corporation-а) започели посао на уједињавању дотадашњих метода односно Booch (Booch '93) и метода OMT-а (Object Method Technology). Те методе су се претходних година развијале и имале су једну од водећих улога у самом развоју објектно-оријентисаних језика.

Годину дана касније је изашла радна верзија овог пројекта под називом Unified Method 0.8. Крајем исте године, наведеним стручњацима се придружује и Ivar Jacobson са својом компанијом Objectory Company, Grady Booch, Jim Rumaugh и Ivar Jacobson оснивају UML.

Постојало је више верзија UML-а, а прве су биле 0.9 и 0.91. Информатичке компаније су подржале UML након изласка верзије 1.0 у 1997. години. До данашњег дана излази нова верзија на сваких пар година (од 1.1 до 1.5).

Велика прекретница се догодила изласком UML-а 2.0 (2003. године) и UML-а 2.3 (2010) који данас омогућава знатно веће бољу флексибилност и прилагођеност свим доменима примене.

У принципу UML 2 је углавном исти као и његов претходник, нарочито у погледу коришћења најчешће коришћене централне функције.

2.3. Циљеви UML-а

Постоји доста циљева UML-а. Први и најважнији јесте да UML буде у режиму опште намене, који сви пројектанти могу да користе. Намењен је да обухвати концепт водећих метода, односно као што је већ поменуто требало је да замени моделе OMT, Booch, Objectory и друге. Намењен је да подржи добре праксе за дизајн као што је енкапсулација. UML има за циљ да се бави актуелним темама развоја софтвера и база података великих размера које раде велики развојни тимови.

2.4. Елементи UML-а

UML садржи три основна елемента а то су:

- Основни градивни блокови,
- Правила за повезивање градивних блокова и
- Општи механизми који се примењују у UML-у

2.4.1. Ствари (things)

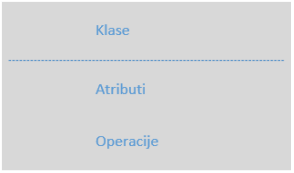
Ствари представљају најважније градивне блокове UML-а. Ствари могу бити:

- Структурне ствари
- Ствари понашања
- Ствари груписања и нотације

2.4.1.1. Структурне ствари

Структурне ствари дефинишу статички део модела. Они представљају физичке и концептуалне елементе. У наставку су кратки описи структурних ствари:

- **Класе** представљају скуп објеката који имају сличне одговорности и понашања.

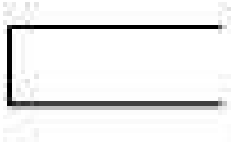


Слика 1. Класе

- **Приступ (Interface)** је у ствари скуп операција којима се утврђује одговорност класе.

Слика 2. Interface

- **Сарадња (Collabration)** –дефинише интеракцију између елемената



Slika 3. Collaboration

- **Компоненте** објашњавају физички део система

Слика 4. Компонента

- **Чвор (node)** се може дефинисати као део физичког елемента који постоји у тренутку извршавања.

Слика 5. Чвор

- **Случај коришћења** представља скуп радњи у извођењу система за постизање одређеног циља.

Слика 6. Случај коришћења (Use Case)

2.4.1.2. Понашања

Ствари понашања се састоје од динамичних делова UML модела. У наставку су представљене ствари понашања:

- **Интеракција** је дефинисана као понашање које је састоји од групе порука које се размењују међу елементима да би се извршио одређени задатак

Слика 7. Порука

- **Стање** дефинише секвенцу стања објекта које пролази као одговор на догађаје. Догађаји су спољни фактори одговорни за промену стања.

Слика 8. Стање

2.4.1.3. Ствари груписања

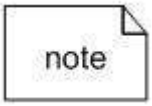
Ствари груписања се дефинишу као механизам који служи за груписање елемената UML модела заједно. Тренутно постоји само један елемент за груписање:

- **Пакети** су једина доступна ствар који служе за прикупљање структурних ствари и ствари понашања.

Слика 9. Пакети

2.4.1.4. Ствари напомене

Анотације односно напомене се користе да изнесу одређени коментар, ограничења и слично једног UML елемента.

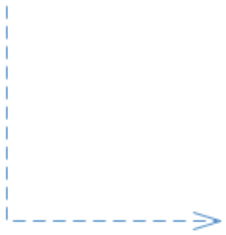


Слика 10. Напомене

2.4.2. Везе (Relationship)

Везе представљају један од најважнијих градивних блокова код UML-а. Помоћу веза се приказује колико су елементи повезани једни са другима и то удружење описује функционалност апликације. Постоји укупно четири врсте веза:

- **Зависност** је однос између две ствари у којима промена у једном елементу директно утиче на другу.



Слика 11. Зависност (dependency)

- **Удружење (association)** је у ствари скуп линкова који повезује елементе UML модела. Он такође описује како многи предмети учествују у том односу.

Слика 12. Association

- **Generalizacija** се може дефинисати као однос који повезује један специјализовани елемент са једним уопштеним. То у ствари описује наслеђивање у свету објеката.

Слика 13. Генерализација

- **Realizacija** се дефинише као однос у коме су повезана два елемента. Један елемент описује неку одговорност која се не спроводи, а други их имплементира. Овај однос постоји у случају интерфејса.

Слика 14. Реализација

2.4.3. UML дијаграми

Дијаграми представљају један од три елеметна градивних блокова UML-а поред ствари (things) и релација (relationships). UML дијаграми представљају крајњи излаз, односно сви елементи који се користе да би се направио комплетан UML дијаграм и он представља готов систем.

Визуелни ефекат UML дијаграма је најважнији део целог процеса. Сви остали елементи се користе да би се направио један такав дијаграм.

UML обухвата девет дијаграма који су детаљно описани у наставку у следећим поглављима:[5]

- Дијаграм случајева употребе,
- Дијаграм класа,
- Дијаграм секвенци,
- Дијаграм сарадње,
- Дијаграм промене стања,
- Дијаграм активности,
- Дијаграм компоненти,
- Дијаграм развоја.

2.4.3.1. Дијаграм случајева употребе

Случај коришћења представља графички приказ функционалности система, учесника, као и њихове везе са случајевима употребе (use-cases).[6] Ови дијаграми омогућавају корисницима да боље разумеју систем и дају поглед корисника на функционисање система, односно на који начин систем ради и како функционише.

Слика 15. Use case дијаграм (дијаграм случаја употребе)

Када је реч о развоју дијаграма случаја (слика број 15) он се темељи на следећим активностима, и то дефинисање: учесника, случајева употребе, типова веза између њих и на крају активност израде дијаграма.

Размевање прве активности- дефинисање учесника захтева најпре појашњење појам учесника у систему.Заправо, корисник јесте човек који користи систем, а учесник јесте специфична улога коју има корисник у комуникацији са системом.Учесник јесте особа или вештачки ентитет (софтвер или систем) који учествује у случају употребе.

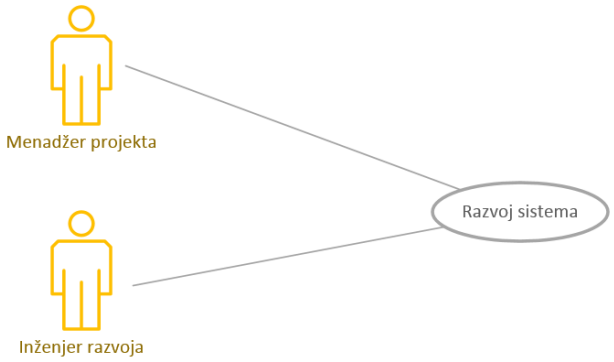
Случај употребе дефинише функционалност система са становишта учесника-шаблон понашања делова система.

Трећа активност се односи на успостављене везе у дијаграму случајева употребе, које могу бити:

- Асоција (Association),
- Асоцијација између типа*include*,
- Асоцијација између случајева употребе типа *extend*,
- Генерализација (Inheritance),
- Зависност (Depedency).

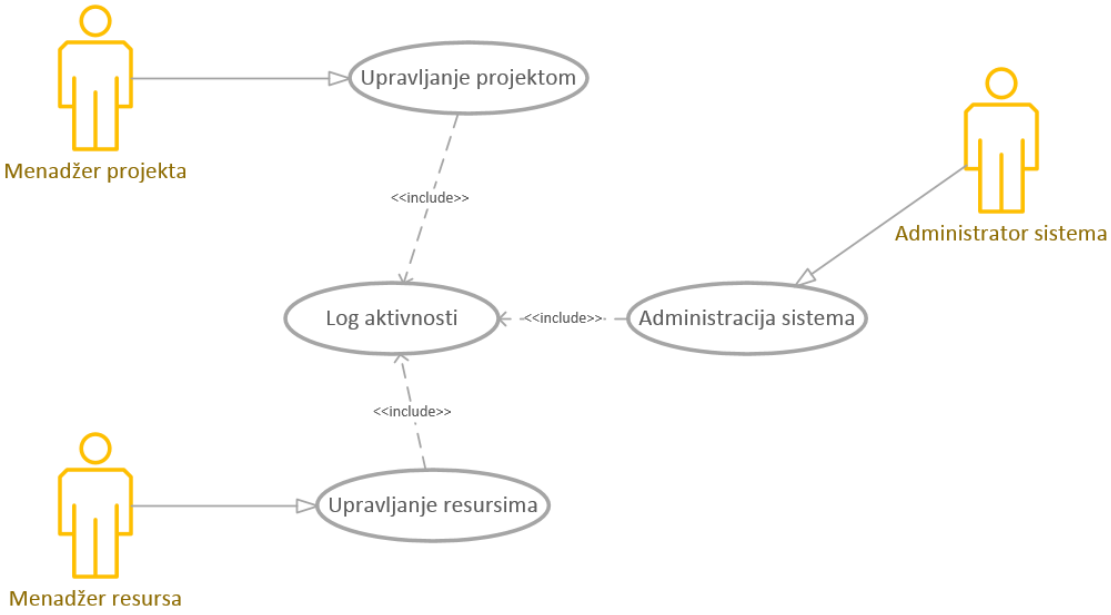
Неке од активности су објашњене у претходном делу, међутим ради бољег схватања биће објашњене и конкретно за дијаграм случаја употребе.

Асоцијација заправо јесте бидирекциона веза - односно линија која учеснике спаја са случајевима употребе. Асоцијација између самих учесника или случајева употребе, дефинише повезаност тих елемената, што се може видети на слици број 16.



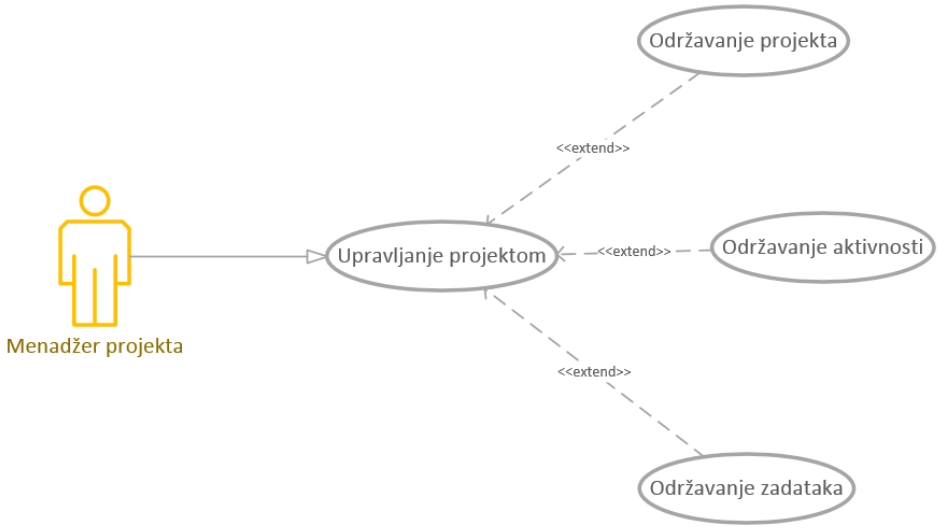
Слика 16. Асоцијација

Веза***include***приказује однос између случајева употребе, у ком један случај употребе користи услуге другог. Пример инцлуде зависности приказана је на слици број 17. Лог активност use-case је главни за управљање пројектом, управљање ресурсима и администрација система усе цасе-ове и укључен је од стране тих усе цасе-ова.



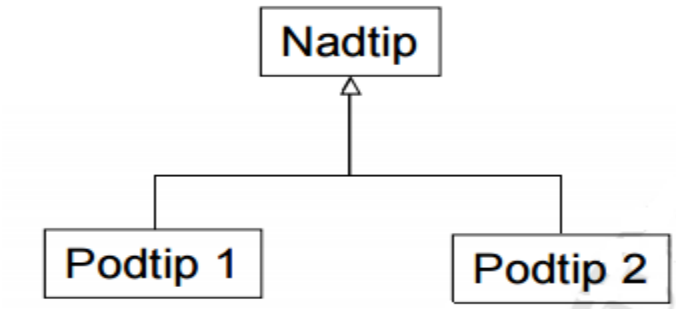
Слика 17. Пример include зависности

Primer ***extend*** зависности приказан је на слици број 18. Одржавање пројекта, одржавање активности и одржавање задатака use-case-ovi представљају опције за Управљање пројектом use-case.



Слика 18. Пример extend зависности

Генерализација подразумева смештање заједничких особина у општу класу која представља надтип, при чему све што важи за надтип, важи и за класу која је подтип.

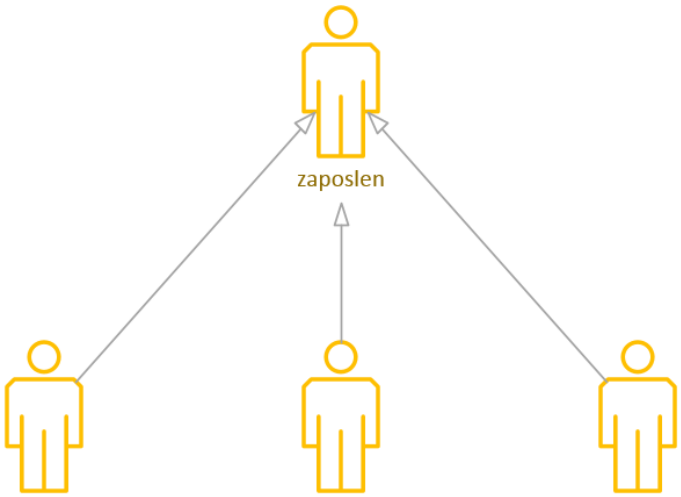


Слика 19. Генерализација

Генерализација се реализује наслеђивањем (inheritance). Важне последице наслеђивања су:

- заменљивост (substitutablility) која омогућава да се уместо надтипа може користити објекат било ког подтипа те класе,
- полиморфизам који омогућава добијање различитих одговора не водећи рачуна о позивању од стране клијента.

Генерализација учесника- изведени учесник има све особине и понашање основног (апстрактног) учесника, али може додати особине или редефинисати понашање.



Слика 20. Генерализација учесника

Генерализација случајева употребе- изведени случај употребе има све особине и понашање апстрактног случаја употребе, али може додати особине или редефинисати ање, слика број 21.

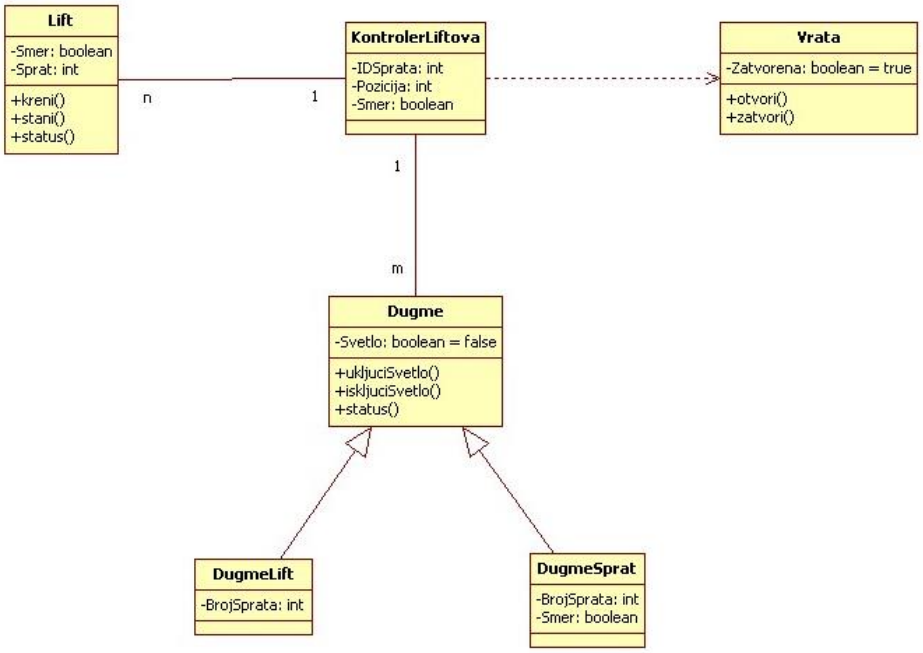
Слика 21. Генерализација случајева употребе

2.4.3.2. Дијаграм класа

Дијаграм класа (*Diagram Class*) јесте основни дијаграм којим се описује структура система. Класа је основни појам у објектном развоју, па самим тим је и основна компонента објектно развијаног система.

Дијаграми класа описују класе система и то кроз опис атрибута који им припадају, операција и релација између класа.

Атрибути представљају обележја која описују одређена својства које класа поседује. Наведени су у оквиру класе, при чему сваки атрибут може имати следећа својства: видљивост, име, тип, кардиналност, подразумевану вредност, мултиплицитет, као и опис неких доругих својстава (нпр. читљивост). Операције описују послове које ће објекат, као конкретна појава класе, знати да обави.Када се од објекта захтева да неки посао обави, то се врши само преко операције коју он поседује. Као и атрибути и операције могу имати одговарајућа својства, као што су: видљивост, име, тип резулатат операције, листу параметара и друге. Релације служе како би се приказао међусобна повезаност класа. Између класа могу да постоје следеће врсте релација: асоцијација (веза са сваком другом), зависност (једна класа зависи/користи се од друге класе), агрегација, генерализација, специјализације (једна класа је специјализација друге класе) и пакета (груписање у једну јединицу тј. пакети).



Слика 22. Пример дијаграма класа

2.4.3.3. Дијаграм секвенци

Дијаграм секвенци (Diagram Sequence) представља интеракцију између објеката у систему, при чему се узима у обзир и временска димензија. Користе се углавном за представњање реализације use-case дијаграма. [7]

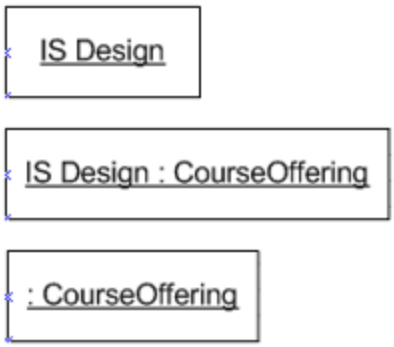
Дијаграм секвенци је графичка илустрација динамичке интеракције где објекти (учесници) комуницирају преко секвенци порука тј.приказује динамичку сарадњу између објеката у времену како би се реализовала одређена операција или добио неки резултат. Када учеснику стигне порука, то значи да он треба да се укључи у неки посао. [8] Овај дијаграм се приказује у две димензије: вертикална димензија приказује време, а хоризонтална одређене објекте.

На дијаграму се приказују поруке које се размењују у дефинисаном временском редоследу између уочених објеката. [9]

У UML- у се објекат у дијаграму секвенци представља правоугаоником који садржи подвучено име објекта. Објекат може бити назначен на три начина:

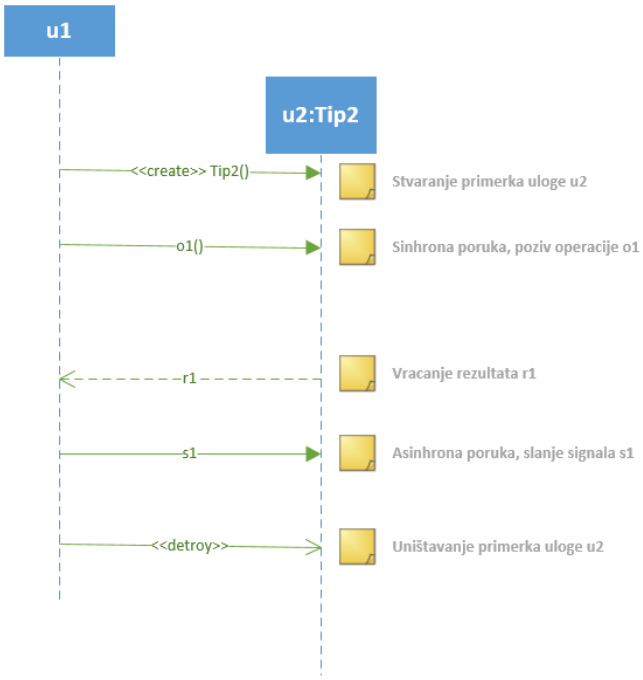
- Само својим именом (имена могу бити посебна),
- Својим именом и класом,

- Само именом класе (анонимни објекат) (користе се да означе било који објекат у тој класи).



Слика 23. Назначеност објеката у дијаграму секвенци

Порука је спецификација комуникације између објеката који преноси информацију, иза које се очекује да следи активност.



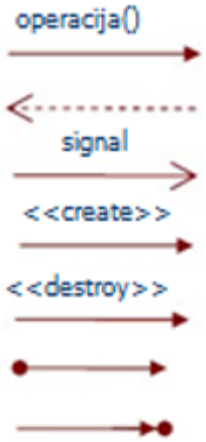
Слика 24. Слање и примање порука

- Порука може бити асинхрона (слање сигнала) и синхрона (позив операције).

- Порука се приказује као стрелица на дијаграму интеракције разне врсте стрелица одговарају разним врстама порука.

UML предвиђа следеће врсте порука:

- позив (call)- покреће операцију улоге примаоца,
- повратак (return)- враћа вредност позиваоцу,
- слање (send)- асинхроно се шаље сигнал примаоцу,
- креирање (create)- креира се објекат (примерак улоге),
- уништавање (destroy)- уништава се објекат,
- пронађена порука (found)- познат прималац, слање није описано,
- изгубљена порука (lost)- познат пошиљалац, пријем неодређен.[10]



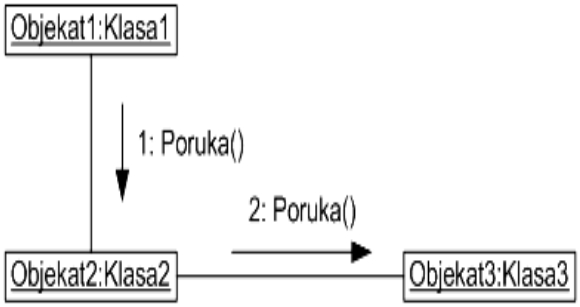
Слика 22. Врсте порука

2.4.3.4. Дијаграм сарадње

Дијаграм сарадње (*Diagram Collaboration*) дефинише комуникацију, па и везе између објеката неопходне за остваривање посматране комуникације. Дијаграм сарадње поред објеката и веза приказује и поруке које објекти међусобно прослеђују, остварујући на тај начин очекивано понашање.

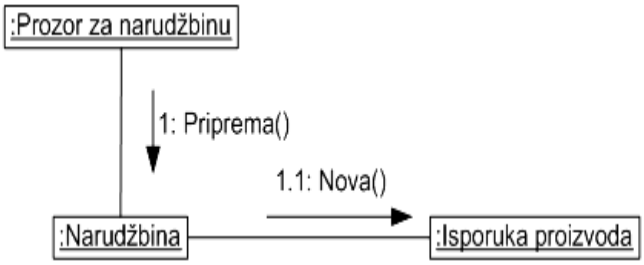
Стрелице између објеката приказују поруке које се прослеђују између њих. Бројеви поред порука представљају секвенцу бројева. Како назив предлаже, бројеви приказују секвенцу порука која се прослеђује између објеката. За једноставније примере може да се

користи секвенца бројева 1.,2.,3. итд., док за сложеније се прелази на 1,1.1,1.2,1.2.1,...итд. На слици број 23 приказани су основни елементи: објекти и поруке које се прослеђују између њих.



Слика 23. Основни елементи дијаграма сарадње

На слици број 24 приказан је једноставан дијаграм сарадње на примеру наруџбине. У овом примеру назив објеката је смештен иза две тачке.



Слика 24. Пример дијаграма сарадње: Наручивање производа

2.4.3.5. Дијаграм промене стања

Дијаграм промене стања (*Diagram State*) је коначни аутомат који садржи стање, прелазе, догађаје и активности. Представља динамички дијаграм који приказује секвенцу стања кроз које објекат пролази током времена (животног века), а као реакција на спољне или унитрашње промене.Опис стања обухвата активности које се извршавају у појединим стањима, акције које се извршавају при преласку из једног стања у друго, као и поруке које условљавају промену стања посматраног објекта.

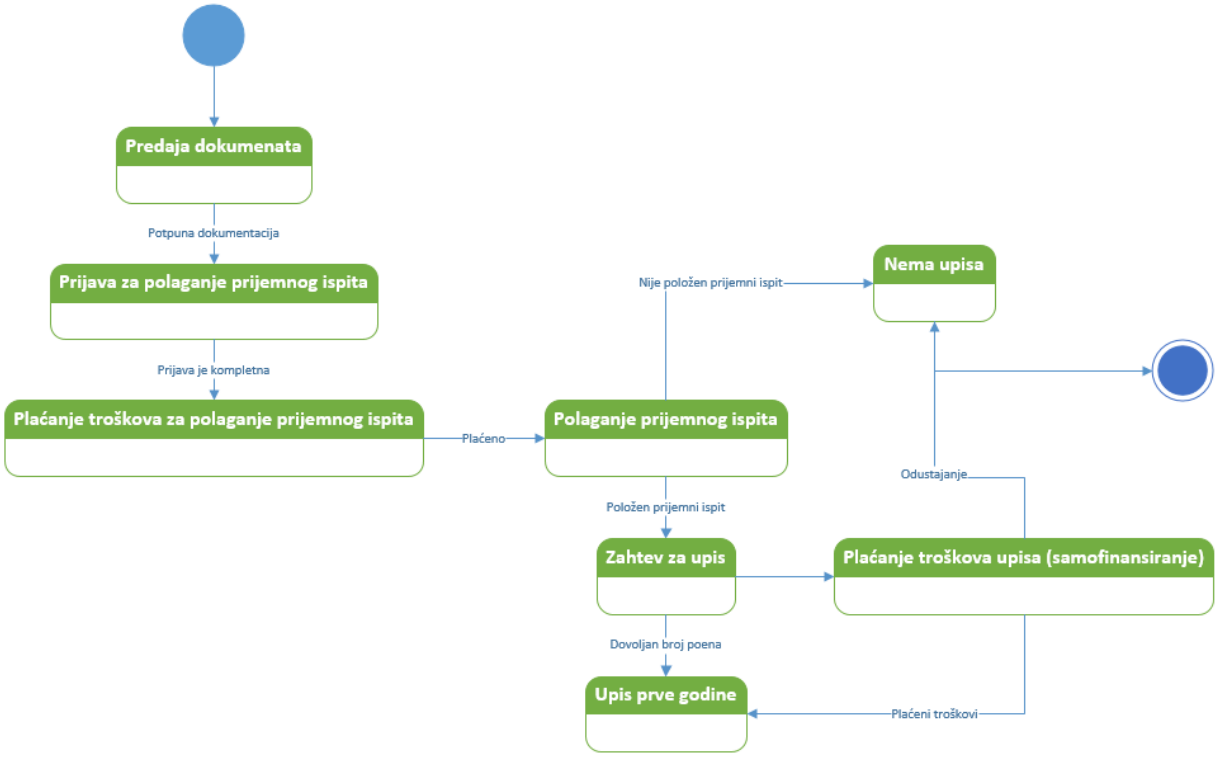
Дијаграми стања појашњавају понашање неког објекта и како се то понашање мења од стања до стања. Такође, показује се који догађај мења стање објекта. Стање је изражено преко вредности атрибута и веза са другим објектима. На пример, стање објекта је:

Сијалица (објекат) је упаљена (стање)

Неки објекат мења стање када се неки догађај догоди тј. када се нешто деси. Ту постоје две димензије енергије, интеракција и унутрашња промена стања. Интеракција описује спољашње понашање објекта и у каквој је он интеракцији са другим објектима (слање поруке или повезивање са другим објектима). Унутрашња промена стања описује како објекат мења стања, на пример, вредности сваког унутрашњег атрибута при различитим стањима.

Дијаграми стања приказују како објекти реагују на догађаје и како они мењају своје унутрашње стање као резултат интеракције.

Дијаграми стања могу да имају полазну тачку и неколико крајњих тачака. Почетна тачка, или почетно стање, представљена је пуним кругом; крајња тачка или крајње стање је представљена малим пуним кругом окруженог великим празним кругом. Стање је представљено помоћу правоугаоника са заобљеним ивицама. Промене стања, или транзиција стања, је приказана помоћу линије на чијем се крају налази стрелица усмерена од једног стања ка другом. Промена стања је именована својим разлогом (догађај који изазива промену стања). Када се догађај догоди, промена из једног стања у друго је испуњена.



Слика 25. Пример дијаграма стања: Упис на факултет

На слици број25приказан је пример процедуре уписа на факултет употребом дијаграма стања. Треба напоменути да на дијаграму постоје два завршна стања, када је факултет уписан и када се одустало због неплаћања трошкова око уписа за самофинансирајуће студенте или ако се пријемни испит није положио.

2.4.3.6. Дијаграм активности

Дијаграм активности (Diagram Activity) приказује секвенцијалан ток активности, односно описује активности које се извршавају у оквиру једне операције. Дијаграм активности је специјална врста дијаграма промене стања којим се приказује ток од активности до активности кроз целокупан систем.

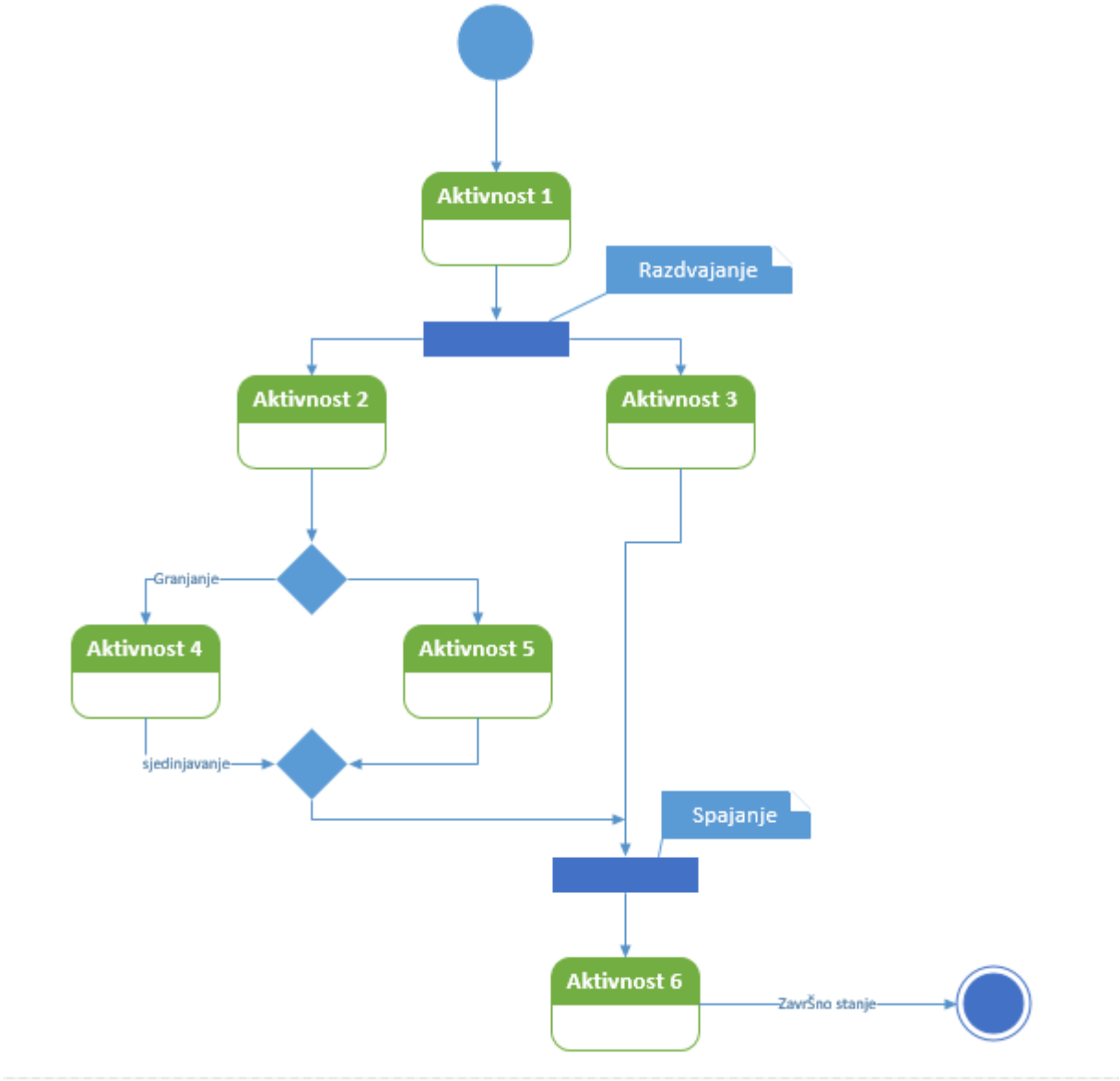
Састоји се од: стања, акција и прелаза и служи за приказ динамичког одвијања пословних процеса.

Акција јесте основна јединица спецификације понашања која представља неку трансакцију или обраду у моделираном систему.Акција је основни извршни елемент активности и представља један корак у активности који се обично даље не декомпонује.

Активност дефинише понашање које се може понављати на више места.Акција се дешава само једном на посебном месту унутар активности.

Дијаграми активности су значајно мењани у свакој верзији језика UML, па су и у верзији UML 2 поново допуњени и измењени.У верзији UML 1 постојали су као специјална врста дијаграма стања.У верзији UML 2, те две врсте дијаграма више нису повезане. [11]

На слици број 26 је представљен пример дијаграма активности, и дијаграми се читају одозго на доле и имају гранање и раздвајање за описивање услова и паралелних активности. Уколико се јави већи број број активности у исто време, користи се раздвајање.



Слика 26. Пример дијаграма активности

На слици број 26 након активности активност 1 следи раздвајање, што значи да се активности: активност 2 и активност 3 одвијају у истом времену, паралелно. Након активности: активност 2 се јавља гранање, које се приказује симболом одлуке (празан ромб). Гранање описује које активности ће да се одиграју након одговарајућег услова. Сва гранања на крају се завршавају и истој тачки која представља сједињавање и означава крај гранања. Након сједињавања све паралелне активности се комбинују и спајају пре завршног стања.

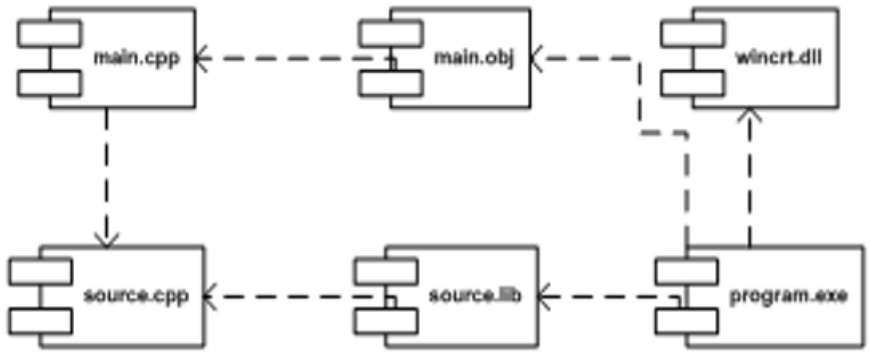
2.4.3.7. Дијаграм компоненти

Дијаграм компоненти (*Diagram Component*) приказује физичку структуру корисничког софтвера (изворни, бинарни и exe код). Дијаграм компоненти приказује организацију и зависност између скупа компоненти, а повезан је са дијаграмом класа због тога што свака компонента указује на једну или више класа, интерфејса или сарадња.

Компоненте представљају физичко паковање модула кода, као што су на пример извори кода фајлова, библиотеке, динамичке компоненте или програми који могу да се изврше. Зависности између компоненти приказује како промене једне компоненте у систему могу да утичу на другу компоненту.

Зависности између компоненти се приказује графички испрекиданом линијом између две или више компоненти. Дијаграм компоненти такође приказује и интерфејсе који се користе од стране компоненти како би се комуникација одвијала несметано између њих.

На слици број 27 дат је пример дијаграма компоненти у коме је показана зависност шест софтверских компоненти.



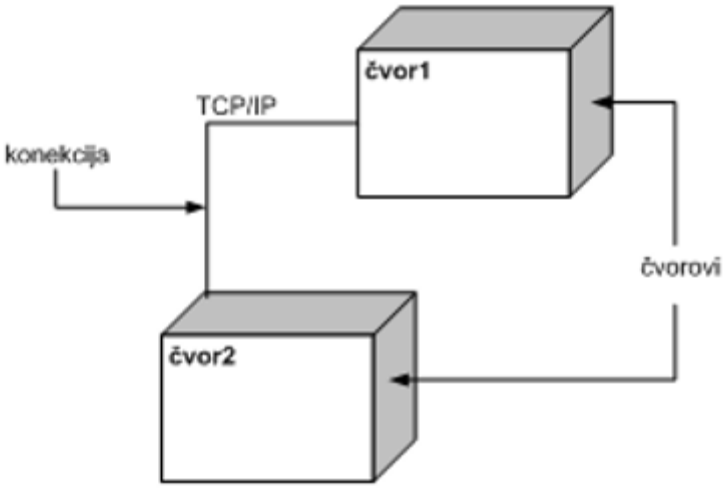
Slika 27.Пример дијаграма компоненти

Дијаграм компоненти и дијаграм развоја се често комбинују у један физички дијаграм. Комбиновани дијаграм комбинује особине оба дијаграма у један.

2.4.3.8. Дијаграм развоја

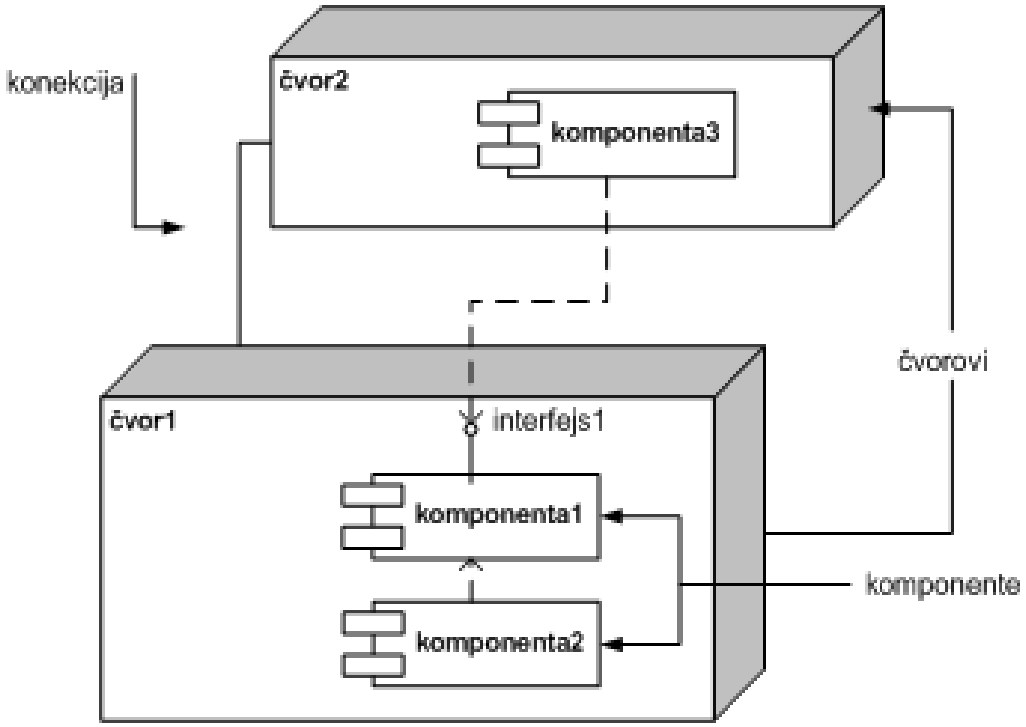
Дијаграм развоја (*Diagram Deployment*) приказује конфигурацију времена процесирања и компоненте у њему. Дијаграм развоја се користи за приказ статичког погледа на архитектуру тј.на физичку структуру хардвера и софтвера.

Дијаграм развоја садржи чворове и конекције.Чвор углавном представља део хардвера у систему (обухвата једну или више компоненти), док конекција приказује комуникациони пут који се користи од стране хардвера за комуникацију и обично указује на протокол као што је на пример TCP/IP.



Слика 28. Приказ чворова и конекције у дијаграму развоја

Комбинован дијаграм развоја и дијаграм компоненти приказан је на слици број 29 и приказује висок ниво физичког описа комплетног система.



Слика 29. Комбиновани дијаграм развоја и дијаграм компоненти

На датом дијаграму су приказана два чвора који заправо представљају два рачунара која комуницирају преко TCP/IP протокола.

Компонента 2 је зависна од компоненте 1 и промене на компоненти 2 утичу на компоненту 1.Такође, у систем је укључена и комонента 3 која је преко интерфејса-интерфејс 1 повезана са компонентом 1. Приказан дијаграм даје брз поглед на систем

3.0. УПОТРЕБА UML-А У ПРОЈЕКТОВАЊУ БАЗА ПОДАТАКА

Имајући у виду да су данашње базе података изузетно опширне и веома комплексне, потребно је да се обезбеди правилно решење како за креирење тако и за размевање самих система, гледано са стране корисника као и креатора. UML као стандард би требао да омогући при креирању база података лакши приступ проблему као и дефнисање веза између појединачних класа података, односно до ефикаснијег процеса и самог тока програмирања, што подразумева уједно и стварање база података. Најбоље решење се постиже ако се моделирање база података врши истовремено са објектним моделирањем.

Пре него што се започне део о самој употреби UML-а за креирање моделирање база података, битно је непоменути да је објетктно-оријентациони приступ подацима драстично другачији од приступа релационе базе података. Најважнија разлика се заправо огледа у томе што објеткна оријентација интезивно користи наследства, док се за ово не може рећи и директно повезати са чистом релационом базом података. Друга ствар, администратор базе података види базу података као коорпоративни ресурс и бави се контролом приступа подацима и њиховим дефиницијама, док се објектно оријентисани програмер бави подацима као основом за дизајнирање програма.

- UML „Модели класа“ се разликују од пословних концептуалних модела ентитет/релација, јер објектно оријентисана заједница није ограничена у одређивању чињенице шта заправо чини једну класу. У свету ентитет/релација, у моделу података где је извршена пословна оријентација, класа може бити веома значајна за пословање, која тежи да држи одређене информације.

Дакле, оно што је битно јесте да UML јесте широко распрострањен и прихваћен као стандард, али да није у диркетној вези и није му директна намена управо креирање база података.

3.1. Моделирање база података

Оно што је битно знати и напоменути јесте да UML поседује низ предности када је у питању дизајнирање база података, а ово су неке предности: [12]

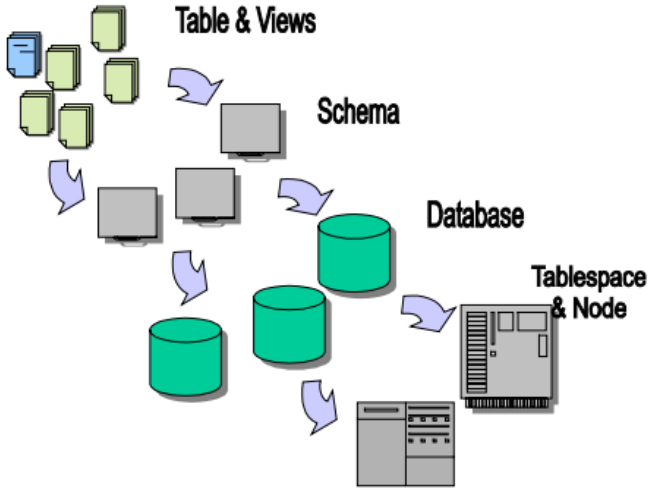
- могуће је обухватање више објеката када се користе дијаграми у односу на традиционалне нотације и
- у исто време се може постићи формирање и обликовање елемената попут домена, ограничења, односа као и табела.Основни елементи за пројектовање база података коришћењем UML-а

Када је у питању дизајнирање базе података коришћењем UML-а, неки од основних елемената који се користе да би се формирала једна целина јесу:

a) **Табеле** – које служе за форимрање групе информација који се сврставају по колонама, и подељене су искључиво према области или теми
b) **Колона** – садржи појединачне атрибуте у табели и саставни је део табеле
c) **Примарни кључ** – је кандидат за кључ који јединствено идентификује сваку врсту у табели
d) **Страни кључ** – представља референционално ограничење између две табеле, односно то је колона или низ колона које се налазе у табели које усмеравају ка примарном кључу који се налази у другој табели
e) **Ентитетски однос** – представља однос две табеле у којој је једна табела подређена другој ако се посматра у хијерархијском смислу
f) **Неентитетски однос** – представља однос две табеле такође, али које су међусобно независне
g) **Поглед** – садржи неке особине табеле, али у целој структури не поседује независтан положај

h) **Складиштене процедуре** – функција која се обично извршава на серверу, и та функција је независна

i) **Домени** – то је низ вредности за сваки атрибут или колону посебно

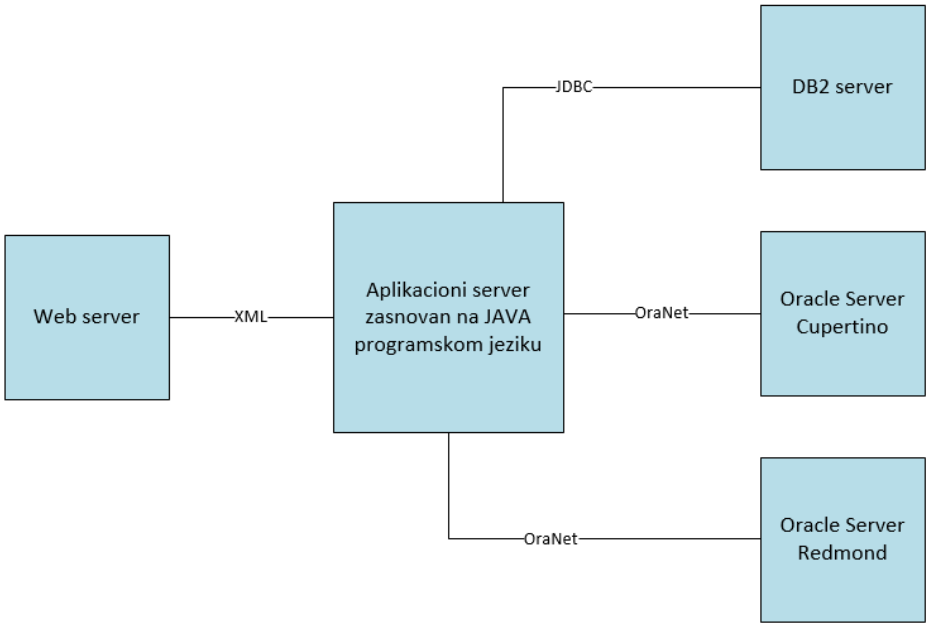


Слика 30. Разноликост имплементације база података

На слици број 30 је представљено разноликост распоређивања одноно имплементације база података. Сложени задаци садрже табеле и погледе, шеме у бази као и чворове, што може додатно збунити администраторе база података. Оно што је заправо битно код администратора јесте да се омогући да на што бољи начин планирају дистрибуцију и конфигурацију саме базе података.

3.1.1. Чвор (Node)

Чвор се може представити у конкретном случају као рачунар, где је смештена база пдатака. То је део основног UML-а. Дијаграм обухвата чворове и везе између чворова. Везе представљају комуникацију, и то је представљено на слици број 31.



Слика 31. Дијаграм распоређивања

Чворови „DB2 server“, „Oracle Server Cupertino“ i „Oracle Server Redmond“ презентујучворове, док XML, JDBC i OraNet комуникационе протоколе. Дијаграм распоређивања је важан код администратора базе података за конфигурацију сервера.

3.1.2. Tablespace

„Tablespace“ представља локацију за чување података, који представља у ствари систем база података. То је веза између физичке структуре која је транспарентна и која се назива База и чвора. Она представља стереотипну компоненту код UML моделирања базе података. Најбоље се може схватити као физички простор за складиштење, који одржава база података. Сама база може бити дистрибуирана на више таблеспаце-а, у зависности од величине података, као и захтевима за приступ подацима али наравно и самих безбедоносних захтева.

3.1.3. База података

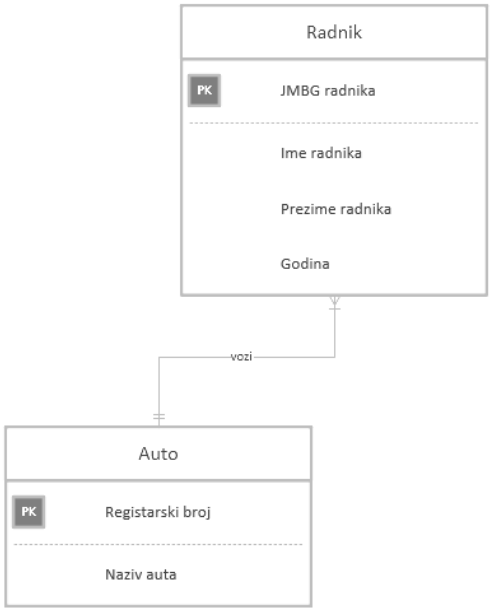
База података представља систем за физичко складиштење података, као и контролу и приступ тих сачуваних података. То је уједно и највећи специјализовани елемент за моделирање података. База података представља такође стереотипну компоненту и део је UML. База података дефинише врсту базе података и ограничења за моделирања података (типове података, ускладиштене структуре, синтаксу итд). База података се користи заједно са другим врстама компоненти у дијаграму, где се дефинишу зависности између апликације и базе података.

Вредности компоненти база података је у значају још код планирања доступности базе података. Додела шеме бази података дефинише основну структуру складиштења информација. Сам администратор базе података користи дијаграм распоређивања како би могло да се омогући препознавање проблема у комуникацији између апликација и база података, и да заједно дефинишу физичко распоређивање података заједно са дијаграмом распоређивања.

Основна јединица организације табела је шема. Шема је исто стереотипни пакет и део UML пројектовање података. Шеме се додељују компонентама базе података на виши ниво детаља. Шеме су организоване у дијаграмима класа. Шему треба прикључити бази података, као што база дефинише ограничења језика, типа података и врсте ускладиштених процедура.

3.1.4. Табела

Табела представља основну структуру моделирања релационе базе података. Табела представља један скуп записа исте структуре, односно истих особина који се другачије називају и редови. Сваки од ових записа треба да садржи податке који се чувају у бази података. Табела је стереотипна класа и део UML моделирања база података. На слици број 32 је представљен дијаграм који показује табеле и везе између њих.



Слика 32. Пример ентитета и њихове везе

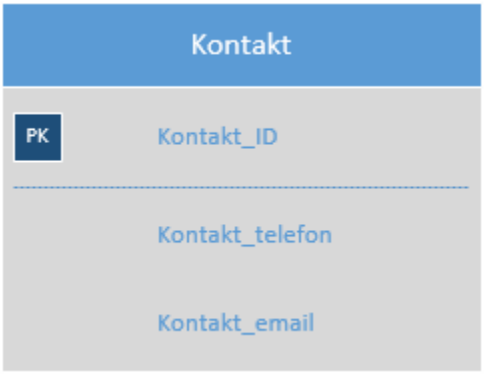
Табеле у UML-у, тј при пројектовању база података играју веома битну улогу. Пре свега велика предност је што је графички веома лепо приказано, и јасно за преглед, па пројектант или администратор има лаку контролу над подацима, док архитекту не занимају сви детаљи о моделу података, али може да провери на једноставан начин да ли су све информације заступљене у бази података.

3.1.5. Погледи

Погледи представљају виртуелну табелу. Поглед (views) заправо представљају скуп записа исте структуре, баш као табеле, само са разликом у томе што је физички извор података у другим табелама. Поглед је стереотипна класа и део UML моделирања података. Погледи су заступљени у дијаграму модела података.

3.1.6. Колоне

Колона је основни организациони елемент унутар релационе базе података. Сваки податак се чува у колони и он је у ствари ред у табели. Колоне су део UML модела података. Колона додаје вредност и тип података који мора бити наведен. Такође, подаци колона могу да буду физички и да се чувају у бази података. Постоје и друге значајне вредности са колонама, који прецизирају детаље о моделу података, као што је „нулл“ и јединственост.



Слика 33. Колоне са атрибутима

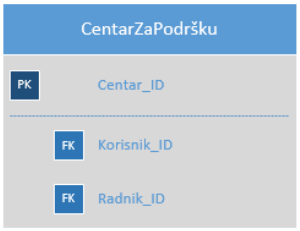
3.1.7. Кључ

Кључеви се користе за приступ табели. Примарни кључеви јединствено идентификују ред у табели. Вредност примарног кључа је најчешће независтан и атуматски генерише базу пдатака како би се олакшало ажурирање података. Страни кључ, увек потиче из односа са другим табелама.

За лако препознавање кључа у колонама, они су означени на следећи начин::

- Примарни кључ PK (Primary key)
- Страни кључ је означен са FK (Foreign Key)
- Уколико се примарни кључ користи у комбинацији са страним кључем користи се ознака PFK

На слици број 34 је представљен пример коришћења примарног и страног кључа.



Слика 34. Примарни и страни кључ при пројектовању база података

Као што је већ речено, кључеви су уједно идентификатори података, и они су неопходни како би се препознала комплетна структура базе података укључујући и све везе између података, како би се добиле информације из чистих креираних делова база.

3.1.8. Индекс

Индекс представља физичку структуру података, и служи како и сама реч каже за индексирање. Индексирање помаже бржем приступу подацима. То је од основних начела објектно оријентисаног приступа проблему, и често се користи при програмирању, а веома је важан елемент при UML моделовању база података. Коришћењем индекса се не мења квалитет података.

3.1.9. Везе

Зависност било које врсте између представљених табела у моделу података се другим речима називају везе. Однос између табела или ентитета мора бити прецизан и тачно међусобно дефинисан. UML класни дијаграм се може употребити за моделовање база података. Ево неколико примера:

1	1	Један према један	Особа - пол
0.. 1	1	Опција на једној страни један према један	Датум смрти - особа
0..* или *	0..* или *	Опционо на обе стране више према више	Особа - књига
1	1..*	Један према више	Место рођења - особа

Табела 1. Пример веза односно кардиналности

А ево и графичких ознака при UML моделовању база података:

Графички симбол	Значење
	Нула или више
	Један и само један
	Нула или један
	Један или више

Табела 2. Ознаке веза код UML моделовања база података у Microsoft Visio 2016

3.2. Софтвери који користе UML за пројектовање база података

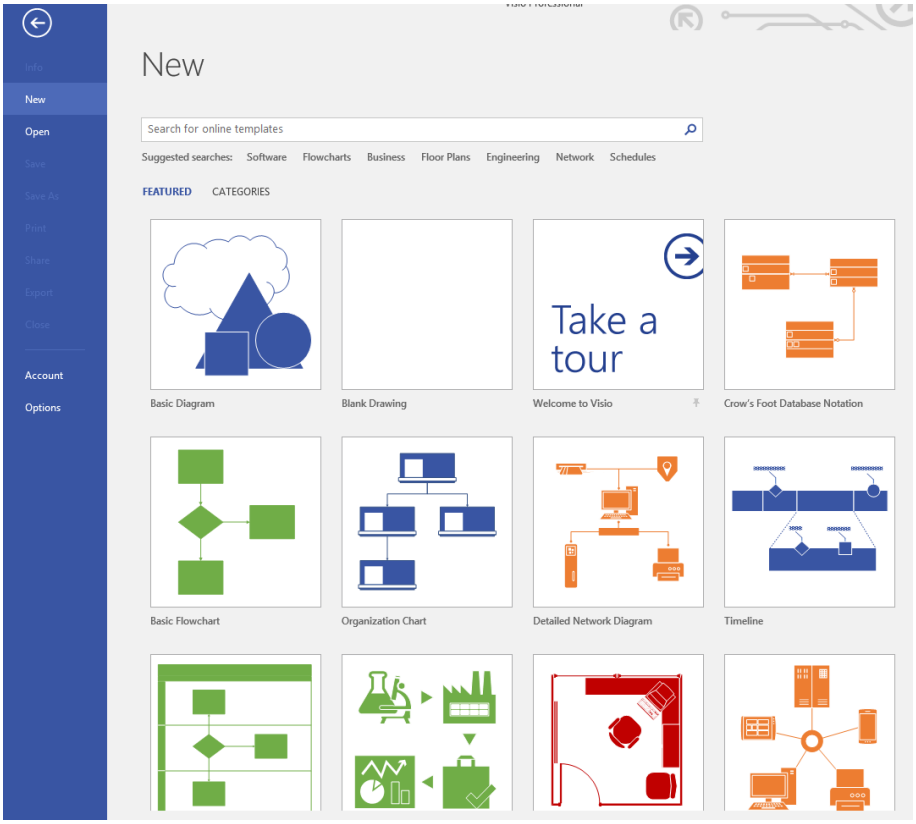
Пројектовање база података се путем UML-а може спровести употребом више софтверских алата, а ово су неки од водећих на светском нивоу:

- Rational Rose XDE (IBM)
- PowerDesigner (Sybase)
- Modelio (Modeliosoft)
- MagicDraw (No Magic)
- Eclipe UML2 Tools (Eclipse Foundation)
- Microsoft Visio (Microsoft)

У поглављу 3.3.1. је објашњено укратко коришћење Microsoft Visia 2016 за потребе пројектовања база података.

3.2.1. Microsoft Visio 2016

Microsoft Visio представља софтвер помоћу којег је могуће израдити дијаграме као и векторску графику у различитим доменима и областима примене. Први производ је настао још 1992. године под Shapeware корпроацијом, да би га преузео Microsoft 2000. године.

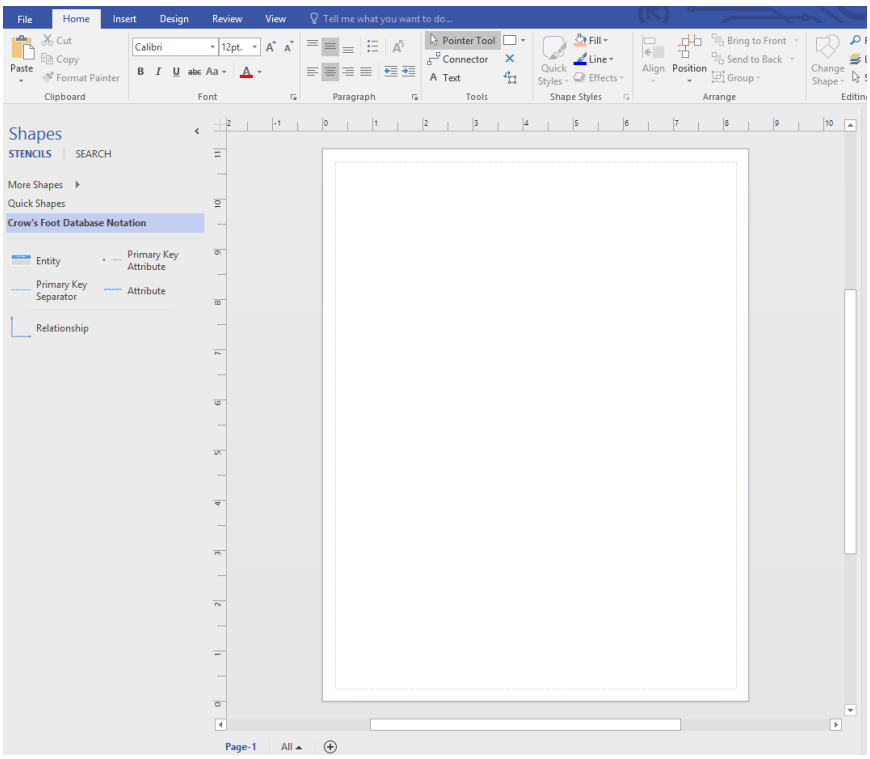


Слика 35. Почетни прозор Microsoft Visia 2016

На слици број 35 се може приметити да Microsoft Visiocаржи велики број темплате-а, и да нуди широк спектар креирања дијаграма. За потребе пројектовања база података најчешће се корсите:

- Crow's Foot Database Notation
- Basic Flowchart
- UML Class
- UML Use Case i drugi

Пошто је Visio врло једноставан за коришћење, биће објашњено укратко коришћење Crow‘s Foot Database Notation-a. Након покретања овог template-a, отвара се прозор као на slici број 36.

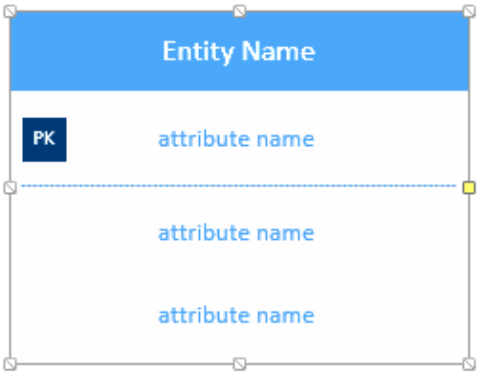


Слика 36. Microsoft Visio

Имајући у виду на једноставност самог софтвера, све се готово своди на коришћење облика, односно Shapes који се налазе са леве стране прозора. Конкретно за овај темплате постоје:

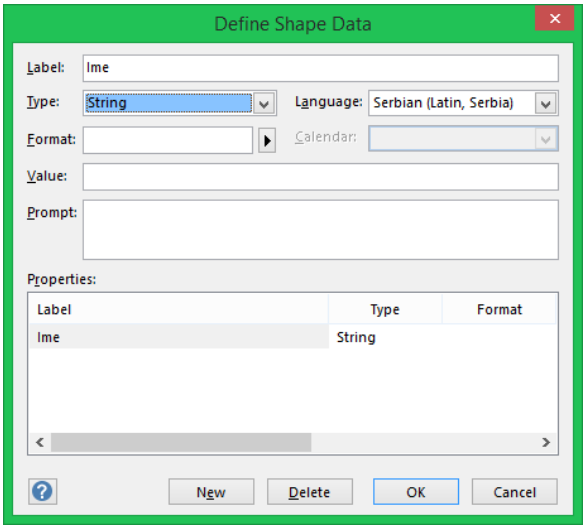
- Entity
- Primary Key Attribute
- Primary Key Separator
- Attribute i
- Relationship

За додавање ентитета односно табела, потребно је само кликнути на Entity и превући на радну површину, након чега се формира сличица као на слици број 37.



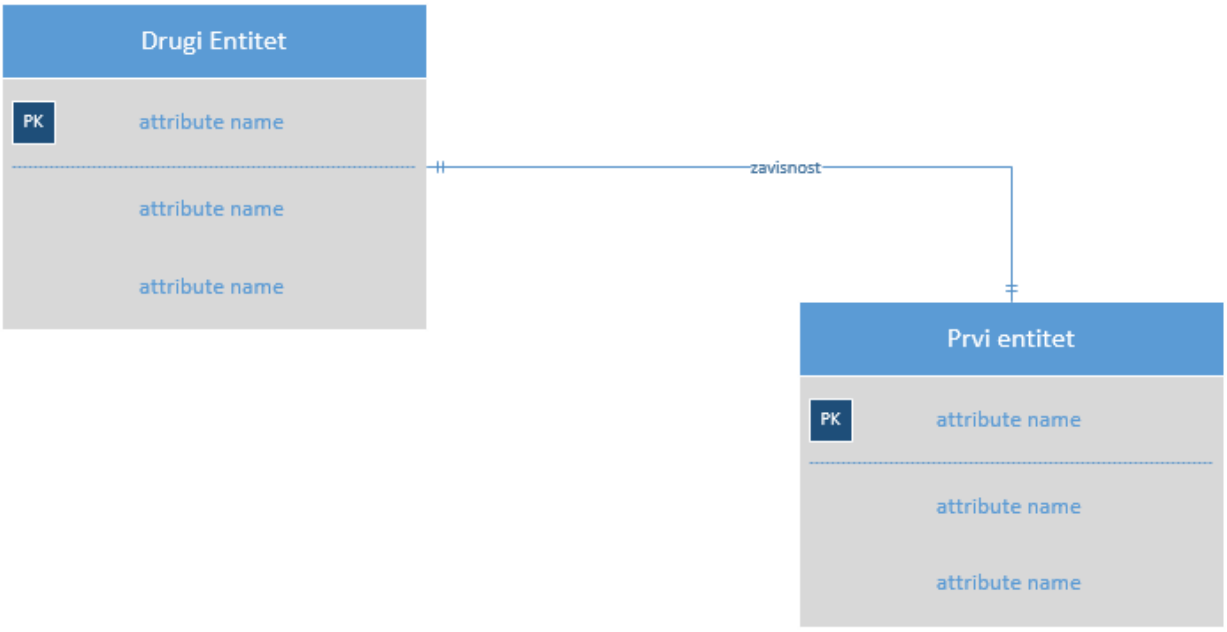
Слика 37. Entity у Visio-у

Аутоматски се генеришу иницијални називи ентитета као и атрибута. Измена назива ентитета и атрибута се мењају једноставним двоструким кликом. Примећује се да је првом атрибуту додељен примарни кључ, који се поставља десним кликом на жељени атрибут, затим „Set Primary Key“, а на исти начин се додаје и страни кључ. Да би се дефинисао тип података за одређени атрибут потребно је такође кликнути десним кликом на жељени атрибут и затим data-define shape data, након чега се отвара дијалог као на слици број 38, где је потребно поставити жељене вредности.



Слика 38. Define Shape Data

Веза између ентитета се такође поставља веома једноставно, кликом на „Relationships“ и превлачењем на радну површиину. Затим се подеси почетак стрелице на први ентитет и крај на други жељени ентитет (слика број 39).



Слика 39. Relationship у Microsoft Visio-у 2016

Овде је представљен однос један према један. Да би се извршила измена, потребно је кликнути десним кликом на стрелицу, односно везу, и одабрати почетак и крај стрелице, баш као што је представљено табели 2.

Као што се примећује, ова врста пројектовања база података је веома једноставна, и решава се само са пар кликова мишем. Такође, веома је лако читљива и лако се разуме, јер је у људској врсти да лакше разуме појаве које су сликовито приказане, што и јесте циљ UML-а.

На сличан и готово исти начин се врши креирање и усе часе дијаграма и свих осталих компоненти који су потребни за пројектовање базе података.

4.0. СТУДИЈСКИ ПРИМЕР СА КОНВЕРЗИЈОМ У ФИЗИЧКИ МОДЕЛ БАЗЕ ПОДАТАКА

У овом делу ће се обрадити део који се односи на само пројектовање базе података које се своди на конкретан пример. Овај пример ће се односити на информациони систем приватне болнице.

За пројектовање базе података фокус је углавном на самом опису базе података, и то обухвата цео процес:

- Опис информационог система
- Опис затеченог стања
- Поставке захтева
- Пословни процеси
- Логичка анализа и
- Физичка ограничења

Битно је још једном поставити велики значај UML моделовања, и велику предност у односу на традиционални начин моделовања базе података.

Традиционално моделовање полази од саме чињенице да је база података основа, али она као таква не може да функционише независно и да постоји само за себе (без клијената и транскација не би било било каквих информација у бази, али и без апликације која има приступ бази не би било доступних података). Помоћу UML-а се пружа могућност да се само са једним језиком моделује целокупни пословни систем, укључујући и апликације, база података и комплетна архитектура система.

У наставку ће бити објашњење, односно пример моделовања информационог система приватне болнице, како би се успешно демонстрирало на који начин се могу како пројектовати како и исправити односно побољшати услуге клијентима (пацијентима).

4.1. Пројектовање информационог система приватне болнице Скористећи начела UML-а

Циљ овог система јесте да се добије један функционални информациони систем болнице, где ће целокупни процес бити аутоматизован, односно где ће се елиминисати потреба да се ручно извршавају сложене операције руковања великог броја медицинских записа. Такође, потребно је ујединити све те податке у једну базу података што подразумева и избегавање пторебе да се ручно подаци преносе из једног облика у други, односно са једног места на друго. Као резултат би требало да произађе побољшање услуге пацијентима захваљујући знатно бољем функционисању самог информационог система, што ће додатно побољшати и рад запослених лекара.

4.1.1. Замисао функционисања приватне болнице

Израдом овог система би требао да обухвати неке основне операције, које подразумевају да лекари и медицинске сестре приступају записима преко рачунара.Рачунари ће бити постављени у свакој одвојеној соби где су пацијенти смештени, као и у канцеларијама.

Сценарио који би се одигравао приликом уласка медицинске сестре у просторију је следећи:

- Медицинска сестра обилази пацијенте како би могла да оцени стање пацијента
- Медицинска сестра приступа подацима пацијента како би видела прописану дијету и терапију
- Прилази рачунару и преко свог личног ПИН-а се логује на систем
- Уколико је ПИН исправан, врши се приступ систему
- Врши се унос имена пацијента и онда се аутоматски приступа свим подацима о пацијенту
- Медицинска сестра прегледа жељене инфромације
- Након одрађеног посла и прегледа информација, медицинска сестра се излогује са система

У претходних 9 корака типичног сценарија, се може приметити да је целокупни систем веома једноставан, али би се њиме знатно смањиле могућности за настајање грешака, и тиме би се повећала ефикасност (мањи број папира којим би иначе располагали), али би се и постигла одлична интеракција између више болница, и слично.

4.1.2. Концептуално моделовање

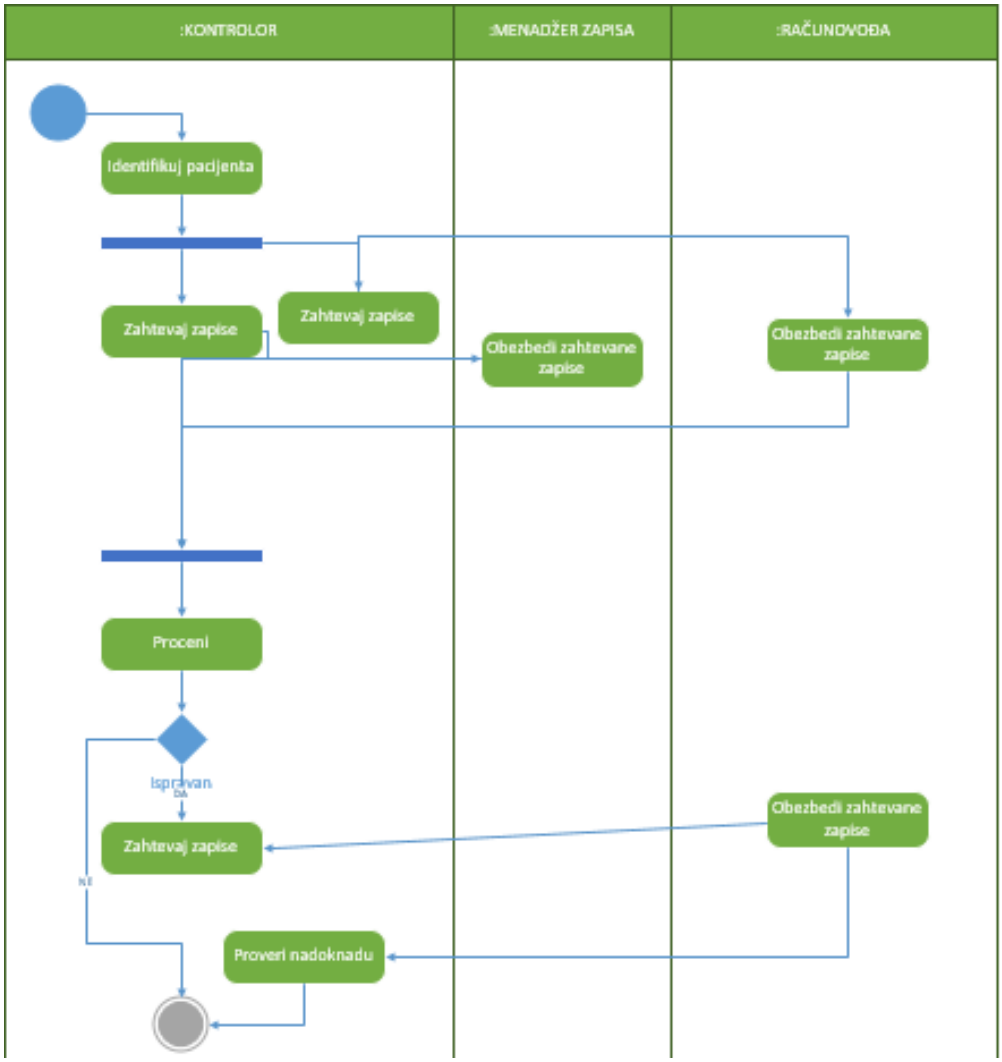
Концептуално моделовање може доста помоћи при моделовању описа који су добијени на основу информација од већег броја особа. UML има велику предност када је у питању разумевање система и моделовања, управо због визуелизације, па је веома погодно да се крене од дијаграма корисничких функција, који уједно представља и почетак, односно неку основу за преглед и идентификацију улога и услуга у болници. Ови дијаграми садрже актере (медицинске сестре, лекаре, пацијенте) и корисничке функције.

На конкретном примеру постоје и актери који нису свакодневни учесници функционисања система а то су: компаније које омогућују превоз пацијената као и неко ко се рецимо распитује за стање пацијента. Постоје и додатне активности попут: Обезбеди болничку негу

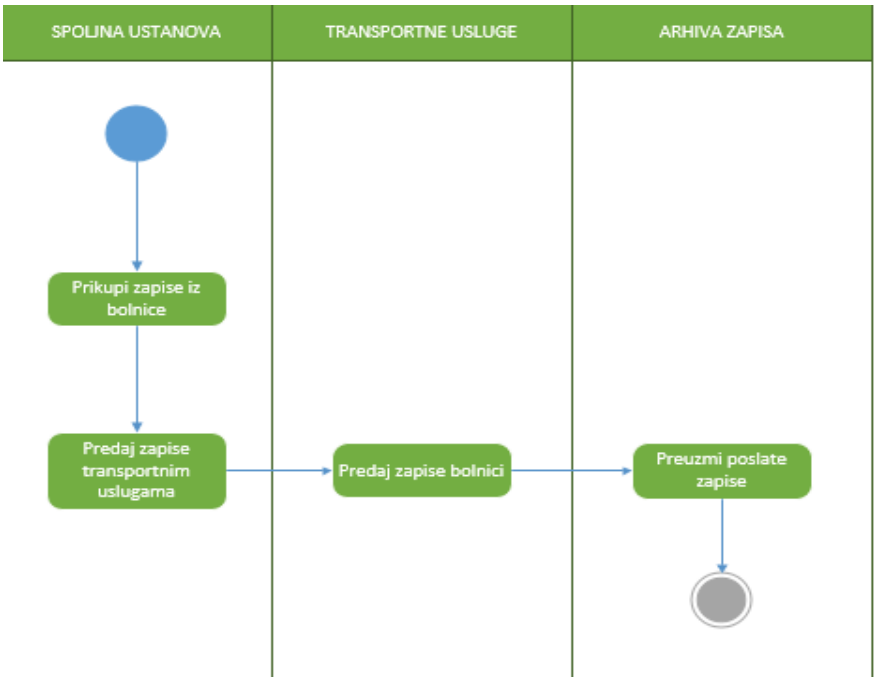
- Наплата и издавање рачуна
- Управљање болничким записима
- Одговори на питања – као одговор на додатне актере који утичу на функционисање система

4.1.2.1. Дијаграм активности болнице

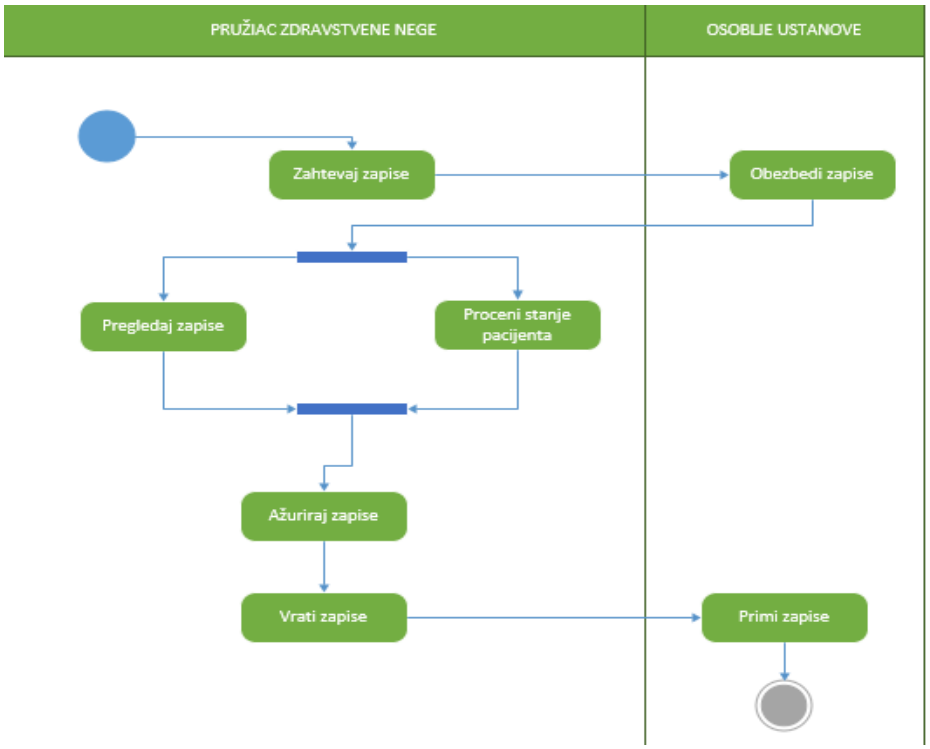
Помоћу дијаграма активности се може врло лако представити начин одвијања послова, и они служе како за разумевање пословног система, тако и идентификацију области тог система, идентификацију активности и евентуалну уградњу информационих потеба одређених активности или пословних корисничких функција.



Дијаграм 1. Дијаграм активности болнице



Дијаграм 2. Дијаграм активности преноса записа



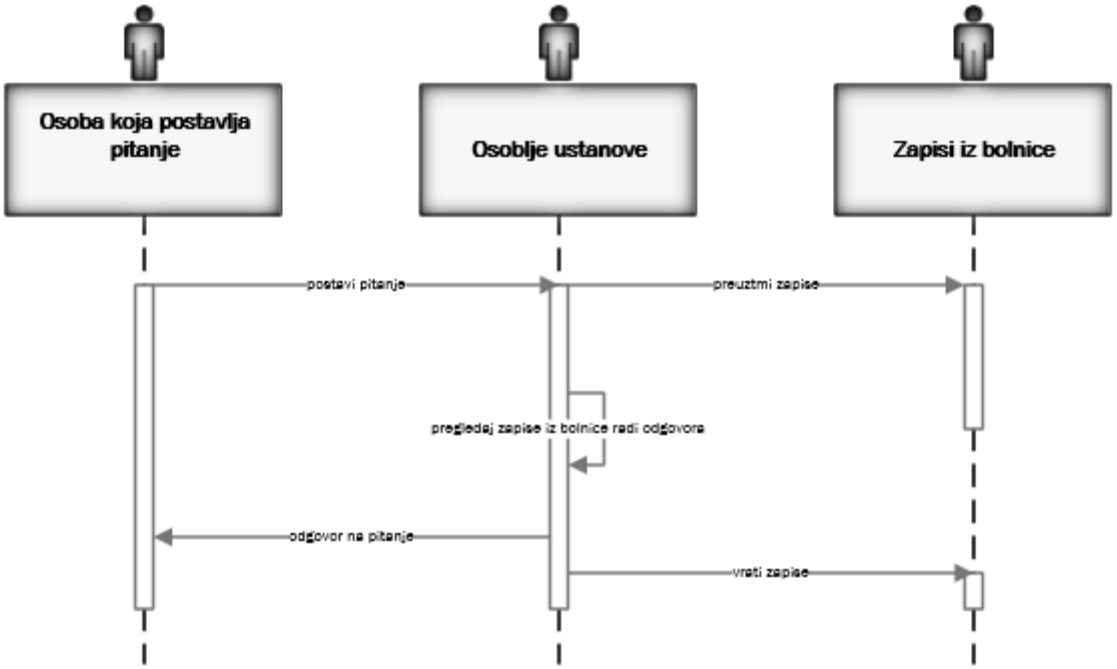
Дијаграм 3. Дијаграм активности обезбеђивање здравствене неге

4.1.2.2. Пословни објектни модел

Пословни објектни модел је заснован на то како људи унутар једног система извршавају пословне процесе, укључујући и интеракцију између њих самих као и ентитета све у циљу дефинисања пословних процеса.

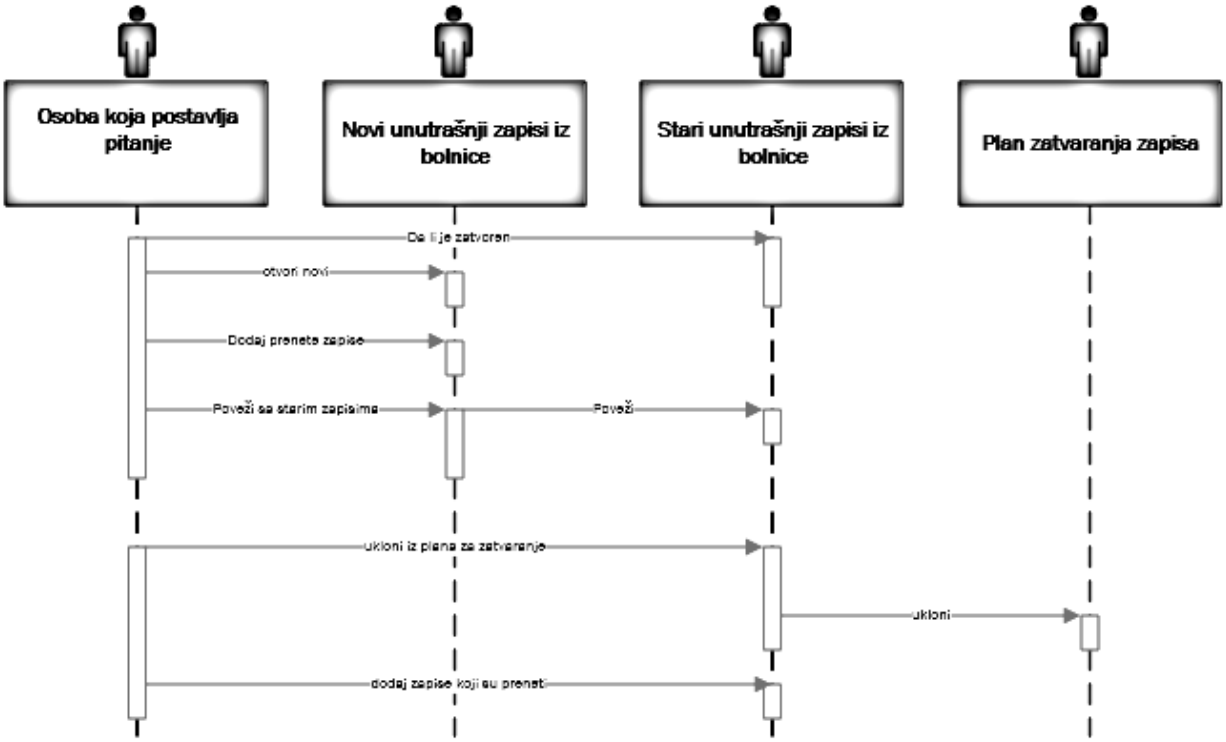
Дијаграм класа је прва компонента пословног објектног модела који садржи актере, раднике у систему, пословне ентитете као и граничне класе укључујући и изузетке и релације.

4.1.2.2.1. Дијаграм секвенци за use case: одговори на питања



Дијаграм 4. Дијаграм секвенци за USE CASE: одговори на питање

Битно је напоменути да се кораци „одговор на питање“ и „прегледај записе из болнице ради одговора“ понављају код сваког питања.

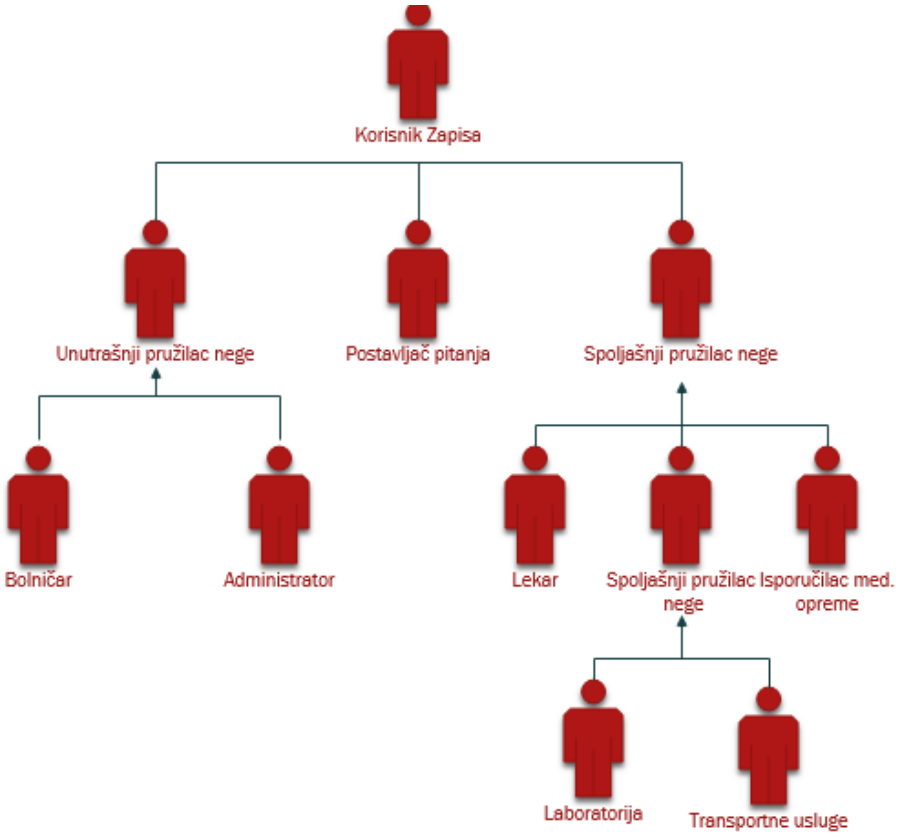


Дијаграм 5. Дијаграм секвенци: прихвати пацијента

Са дијаграма број 5 се може закључити да до коначног плана затварања записа се мора проћи кроз активности које обавља особа која поставља питање, затим се формирају нови унутрашњи записи из болнице, врши се повезивање са старим записима, и на крају се затварају записи, преко методе „уклонити“.

4.1.2.3. Генерализација актера који учествују у систему за обезбеђивање неге у оквиру болнице

Оно што је познато јесте да многи пословни ивзођачи и радници у систему извршавају многе једнаке функције. Ствари се поједностављују увођењем генерализације и они су део у структури генерализације.



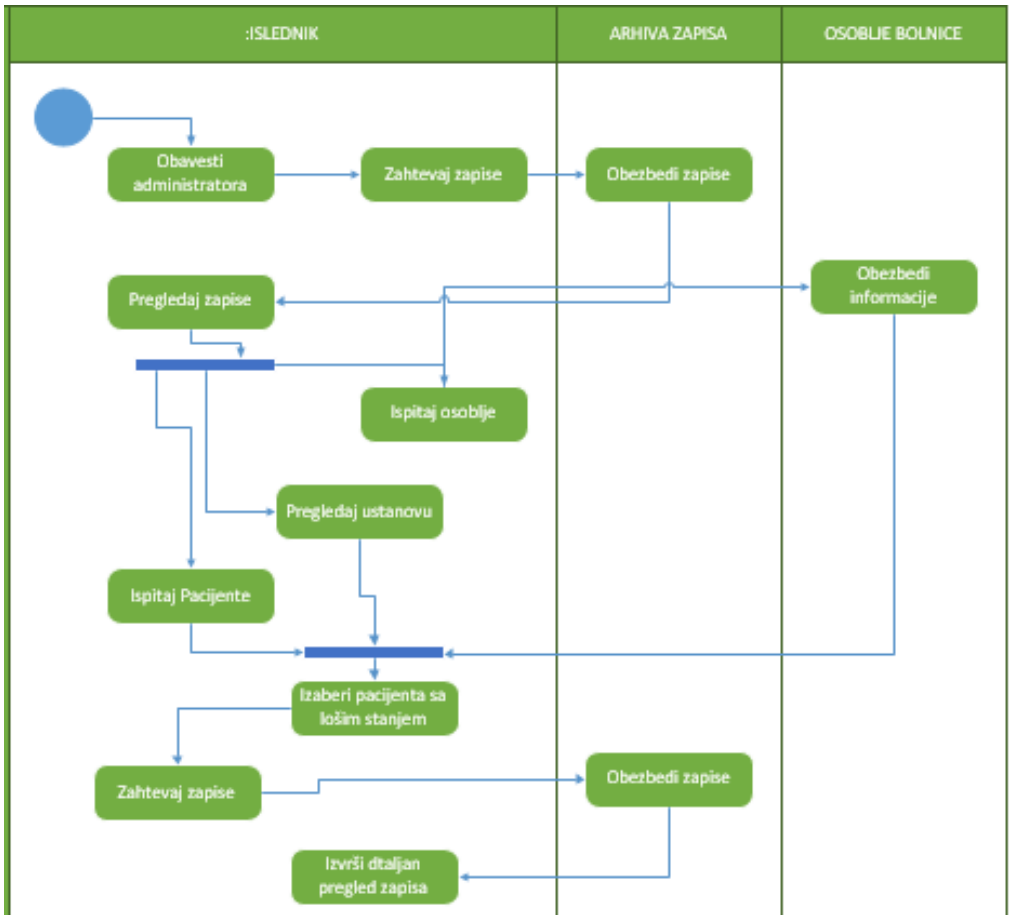
Дијаграм 6. Генерализација актера у систему обезбеђивања неге

4.1.3. Логичко моделовање – дефинисање захтева

4.1.3.1. Дијаграм активности за преглед усаглашености

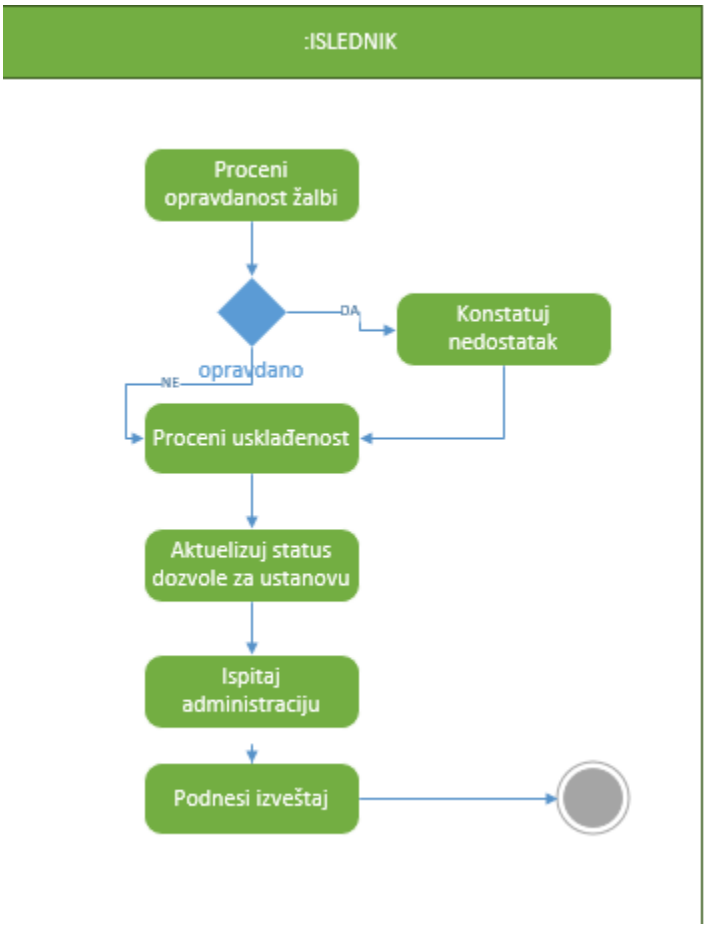
Код дијаграма активности број 7, пролази се кроз три фазе:

1. Иследник – обезбеђује записе архиви записа преко скупа операција
2. Архива записа – прослеђивање записа иследнику, где се врши преглед
3. Особље болнице - обезбеђује информације које се прослеђују иследнику



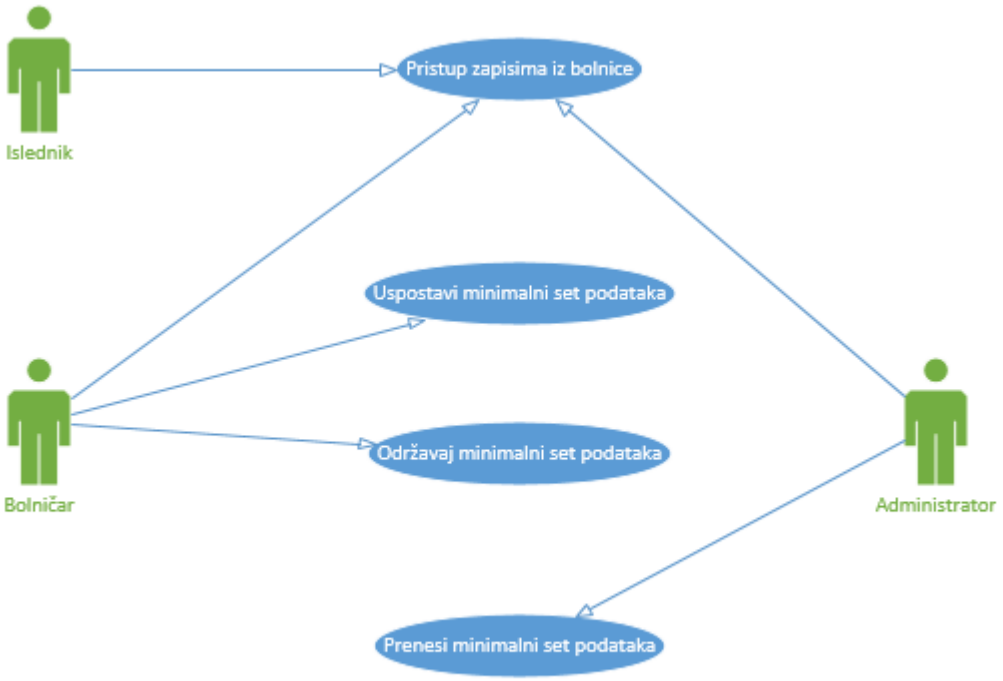
Дијаграм 7. Дијаграм активности за преглед усаглашености

4.1.3.2. Дијаграм активности за процену усаглашености



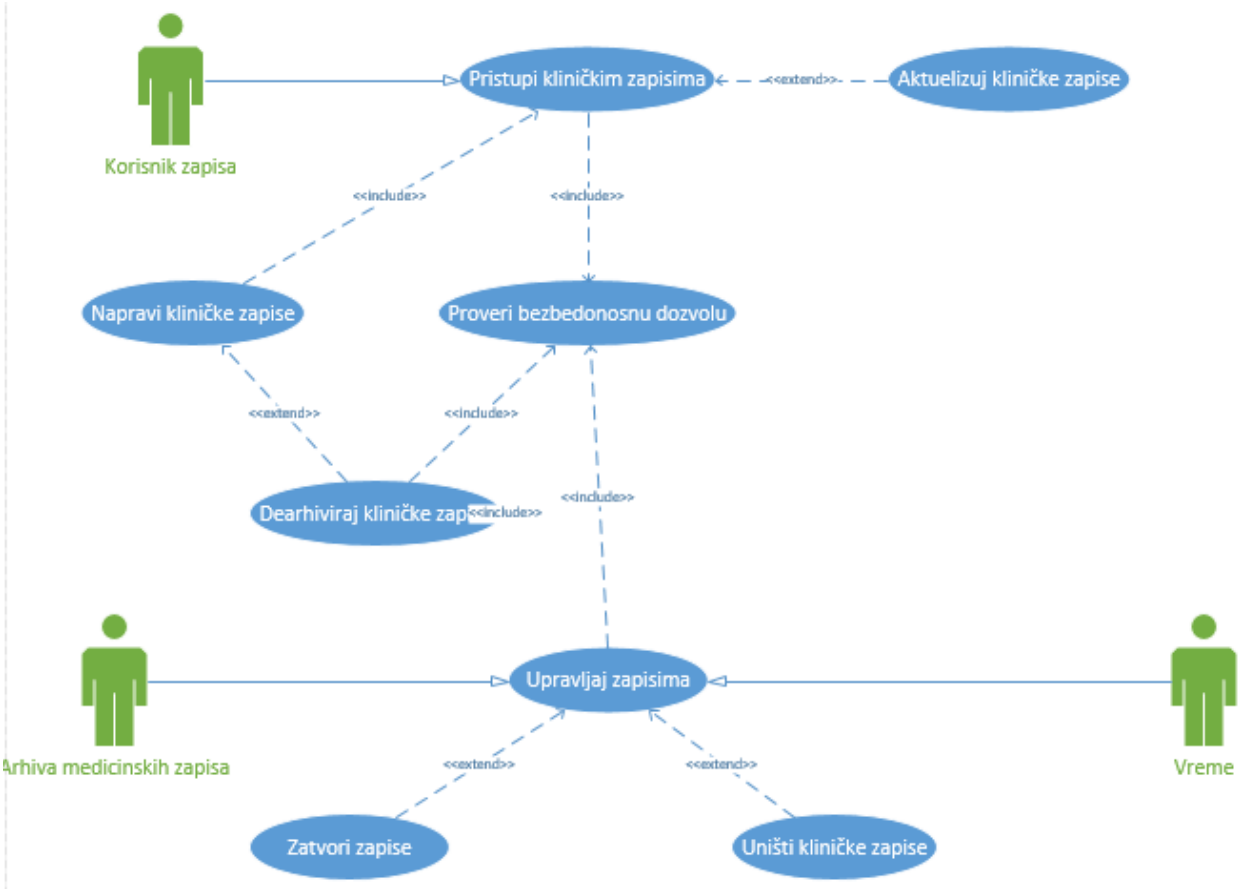
Дијаграм 8. Дијаграм активности за процену усаглашености

4.1.3.3. USE CASE модел подсистема за усклађење са регулативом



Дијаграм 9. Use case модел подсистема за усклађивање са регулативом

4.1.3.4. Модел корисничких функција подсистема за управљање записима у болници



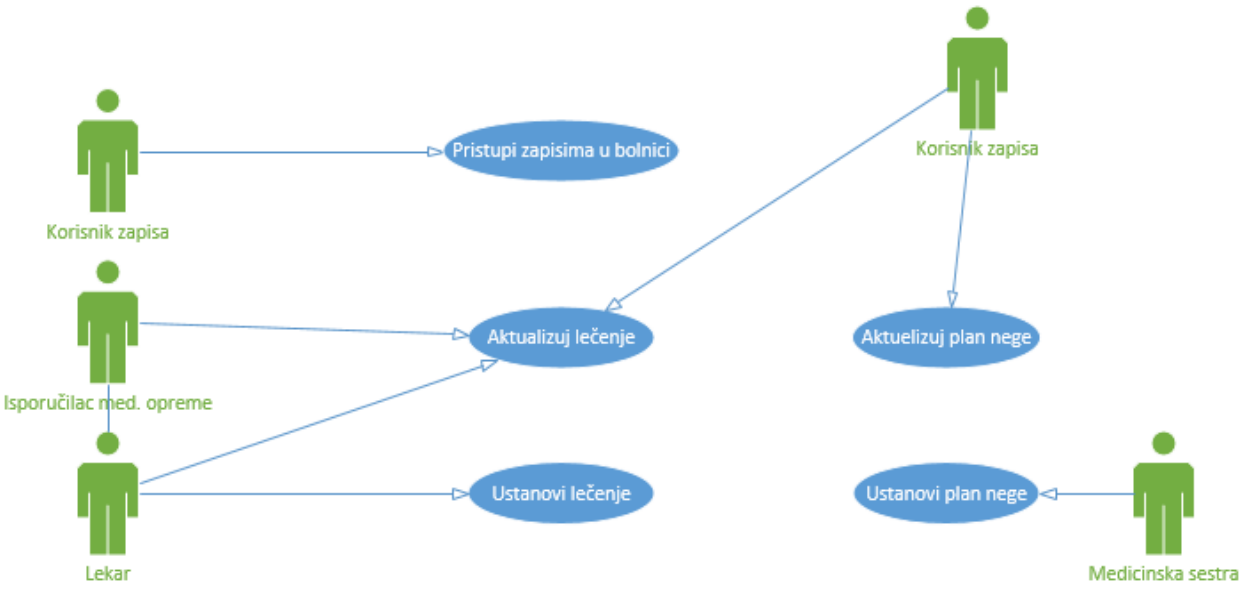
Дијаграм 10. Модел корисничких функција подсистема управљања записима у болници

Са дијаграма 10 се може видети нов актер који се назива „време“, а он је постављен ради стицања осећаја о протицању времена, јер је оно критично у неим функцијама.

4.1.3.5. Use case модел подсистема за обезбеђивање клиничке неге

Секвенца	Приоритет	Итерација	Образложење
Установи план неге	Висок	Прва	Критична
Актуелизуј план неге	Висок	Прва	Критична
Установи лечење	Висок	Прва	Критична
Актуелизуј лечење	Средњи	Друга	Критична
Обезбеди услуге	Низак	Друга	Углавном спољашња

Табела 3. Приоритети секвенци у објектном моделу за обезбеђивање неге



Дијаграм 11. Use case модел подсистема за обезбеђивање болничке неге

4.1.4. Физичко моделовање – припрема трансформације

Код пресликавња, односно трансформације модела, не врши се пресликавање целог модела, већ само елемената модела: класе у табеле, атрибуте у колоне, типове у типове података и асоцијације у релације итд.

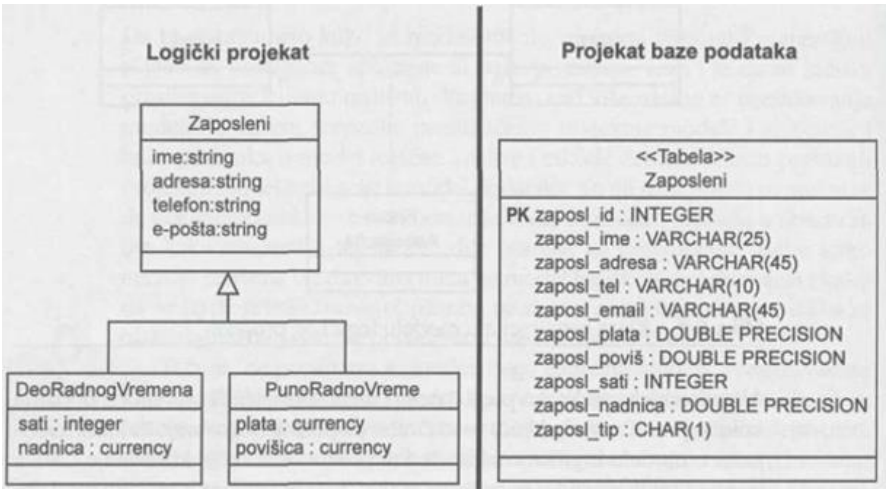
Када се одређене класе подтипова трансформишу у табеле, постоје три избора и то:

- Једна табела по једној класи, односно свака класа се директно пресликава у одговарајућу табелу

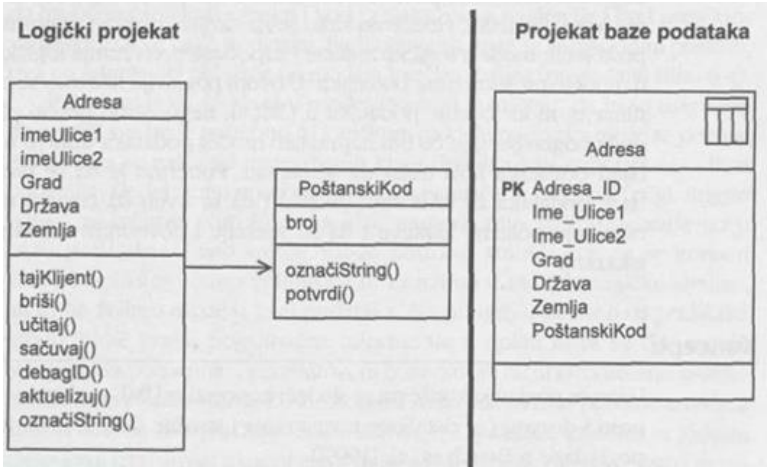
- Једна табела по конкретној класи или такозвано спуштање табела супертипова у њене подтипове

- Једна табела по хијерархији односно подизвање подтипова у супертип, односно нпр. Узме се атрибут из класа подтипова и трансформишу се у колоне једне табеле у коју се преликавају и супертип и подтипови

У конкретном примеру, класа под називом „ЗапосленИУБолници“ има подкласе ПуноРадноВреме и ДеоРадногВремена.



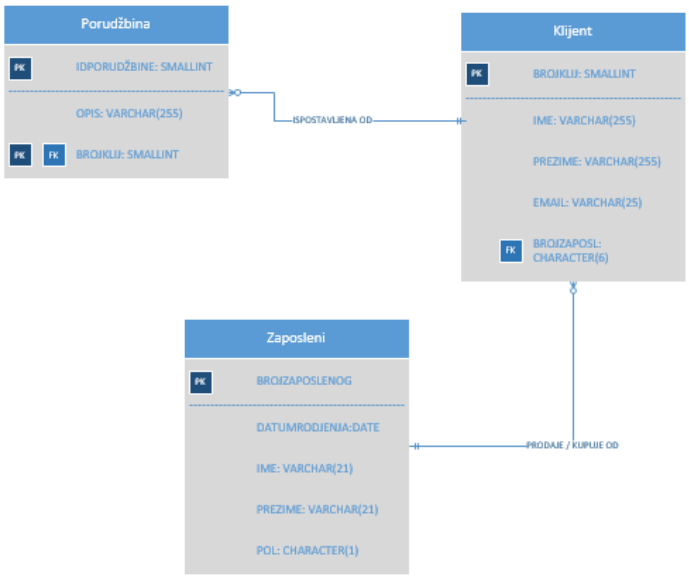
Слика 40. Пример трансформисања хијерархије наслеђивања из дијаграма класа у једну табелу по хијерархији



Слика 41. Пресликавање атрибута у колоне

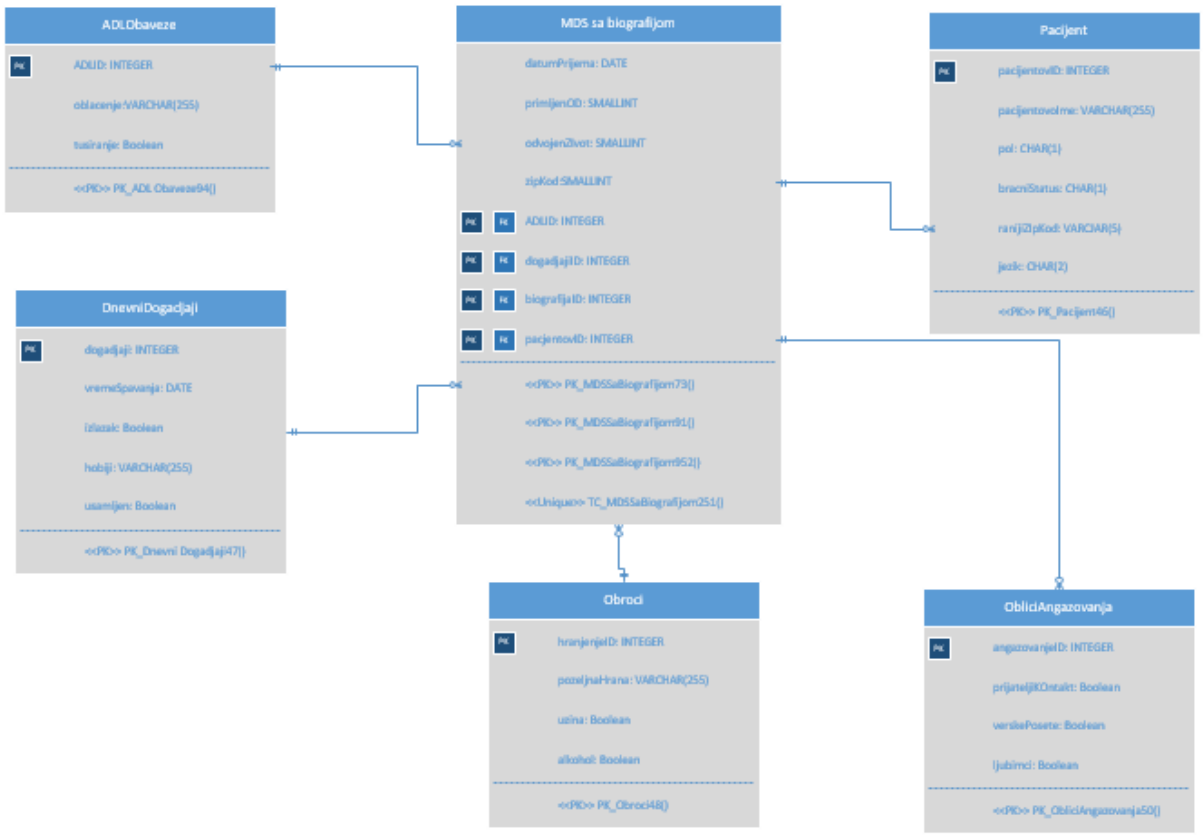
4.1.4.1. Пројектовање базе података

У овом делу, односно пројектовању долази до нормализације логичког пројекта, односно припреме за пројекат базе података.



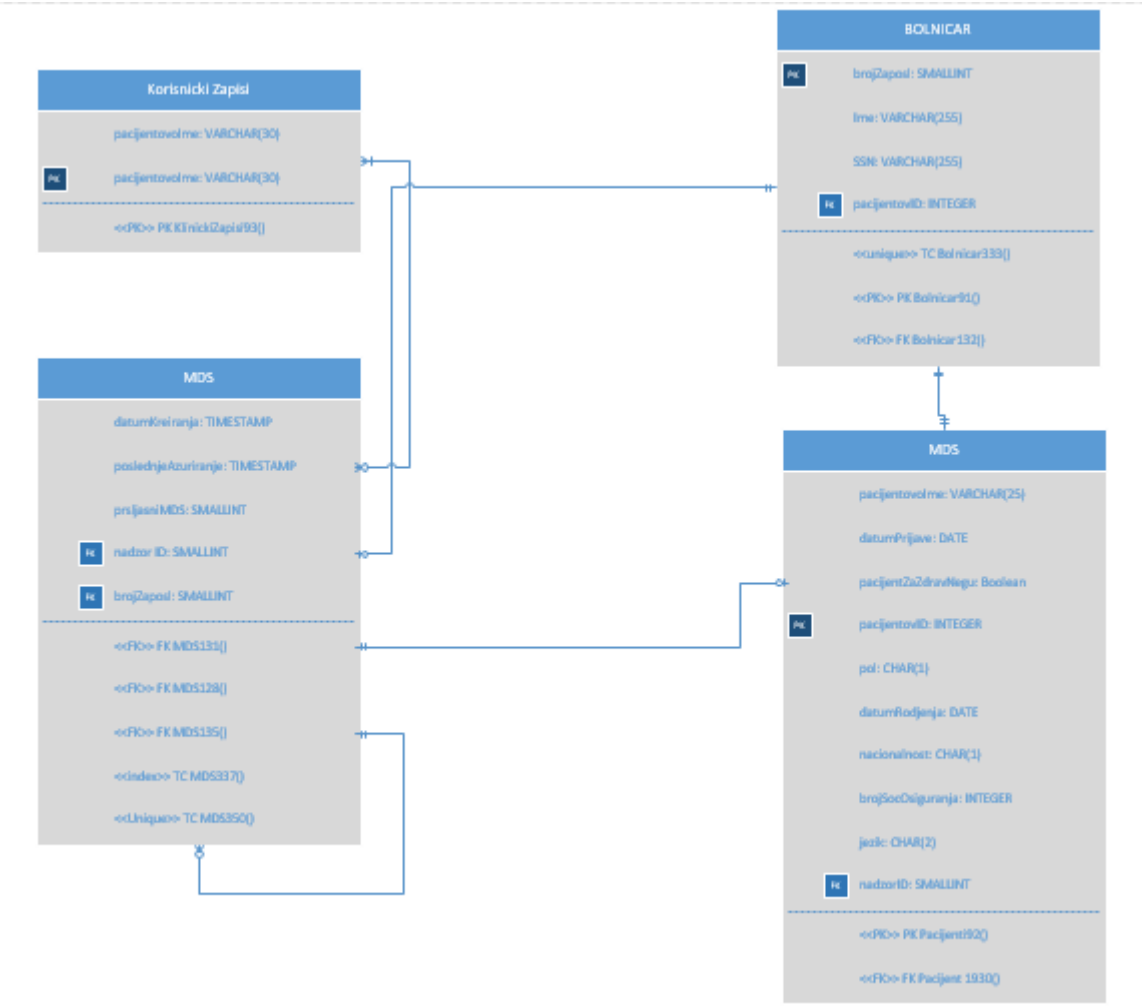
Слика 42. Табеле „Запослени“, „Клијент“ і „Поруџбина“ са њиховим релацијама

Основне биографске информације добијене по пријему представљају у ствари скуп информација прикупљених о пацијенту у полазном МДС-у. Дијаграм 12 представља дијаграм базе података за основне информације које се прикупљају при пријему пре додавања ограничења.



Дијаграм 12. Дијаграм базе података за основне информације које се прикупљају при пријему

Дијаграм 13 представља дијаграм базе података са аспекта болничара.



Дијаграм 13. Дијаграм базе података са аспекта Болничара

Сада је потребно представити дијаграм са потпуним подацима о проценама, и он служи да би се комплетирале све инфор мације у различитим типовима МДС-а односно МДС за комплетну процену. Пројектанти посматрају ситуацију тако што разматрају логички модел и одлучују шта ће у пројекту базе података да трансформишу у табеле и колоне. МДС за комплетну процену садржи велики број табела, колона и релација, па се зато разврставају дијаграми за комплетну процену да би се олакшало читање и његово разумевање.

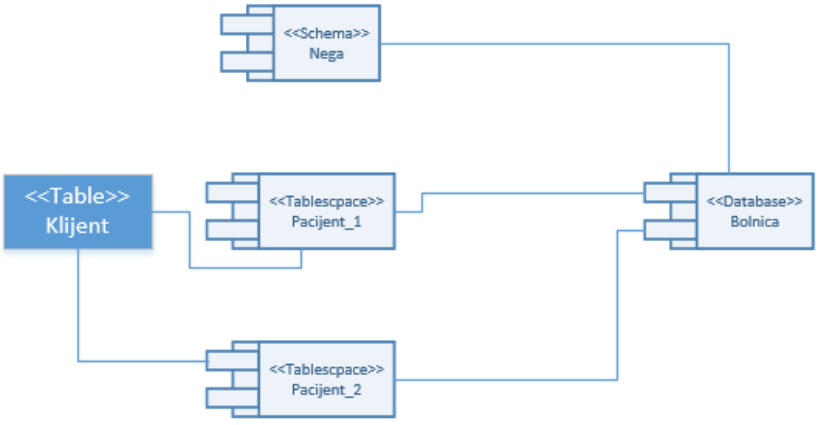
Дијаграм 14. Табеле које се појављују само у МДС-у за Комплетну Процену

4.1.4.2. Реализовање физичких аспеката

Пројектанти при пројектовању база података се баве:

- Величином базе података
- Смештајем базе података (хардверски и софтверски гледано)
- Партиционисањем података
- Специфичним својствима изабраног система

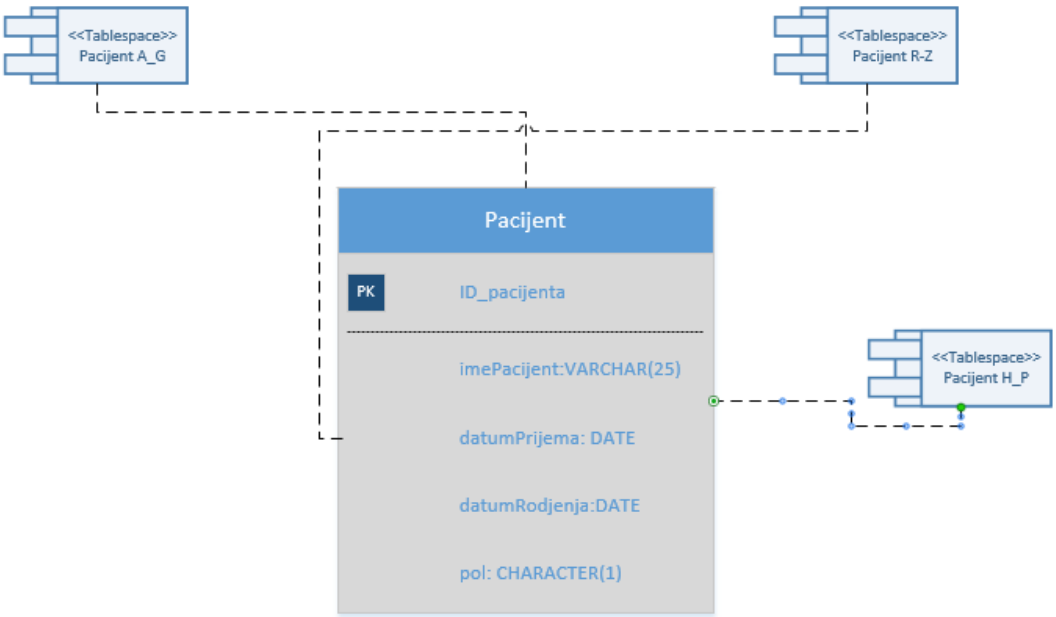
4.1.4.2.1. Елементи који се користе за пројектовање смештаја



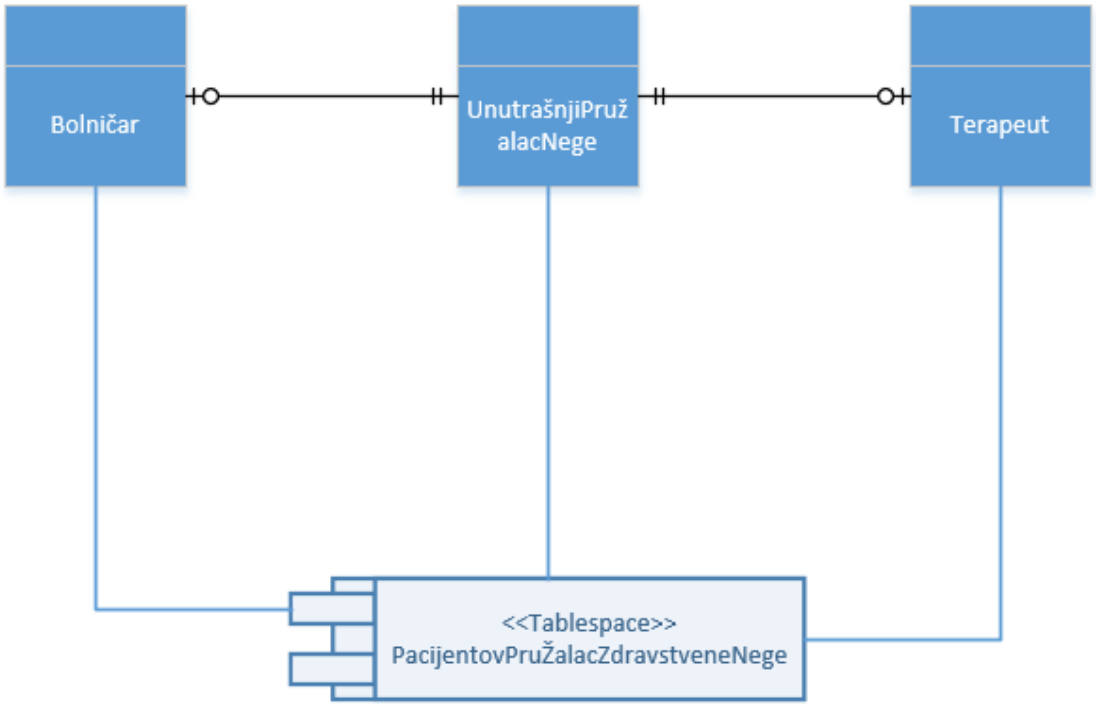
Дијаграм 15. Елементи који се користе за пројектовање смештаја

На дијаграму број 16 је представљен пацијент који је патрицонисан у три простора табела. Табела „Пацијент“ би требало да буде веома велика табела јер садржи све информације о свим пацијентима из клинике. Из тог разлога пројектант је одлучио да их подели у три простра табела, и то на основу слова њихових презимена: од А до Г, од Х-П и од Р-З.

Када се табеле посматрају на те начине, саме табеле су много прегледније, и омогућавају брже и лакше упите у исте.

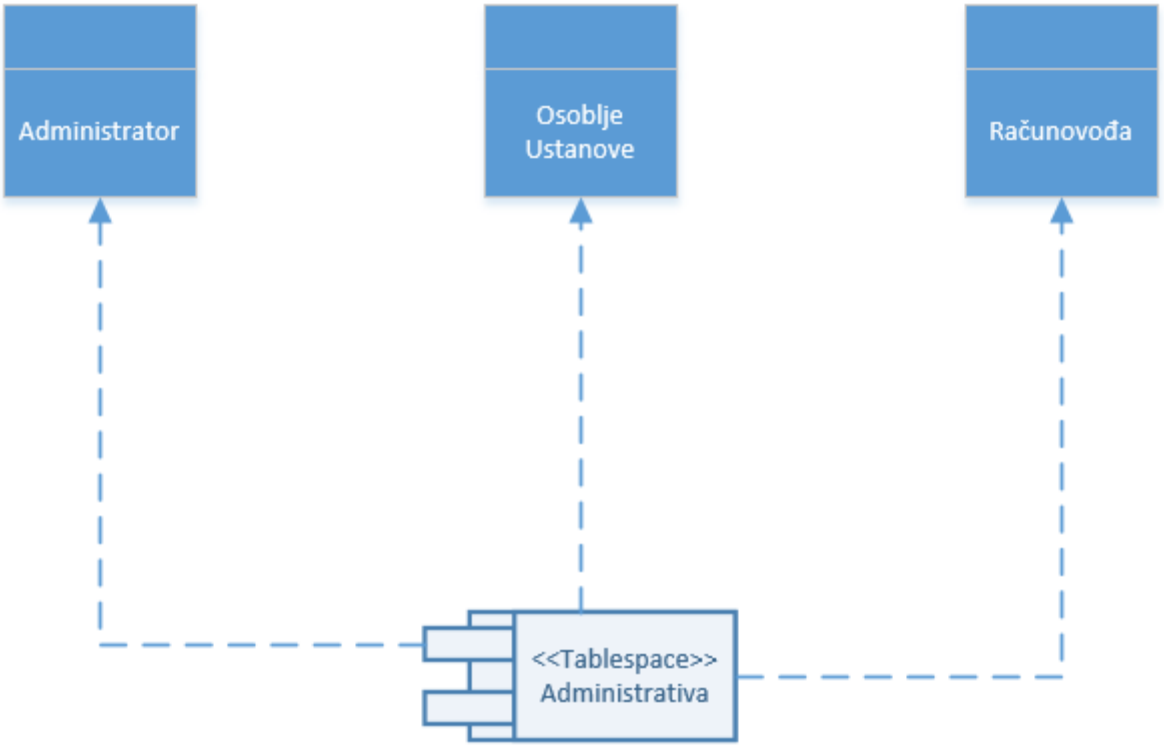


Дијаграм 16. Табела пацијент патрицонисана у три простора табела



Дијаграм 17. Простор табеле „пацијентов пружилац здравствене неге“ и његове зависне табеле

На дијаграму 17 се може уочити да пројектант гледа табеле највишег нивоа и открива да има неколико врста пружилаца здравствене неге (Болничар, Терапеут и Унутрашњи пружилац неге) и додељене су у једну заједничку табелу.



Дијаграм 15. Простор табеле Административа и његове зависне табеле

Као што се види са дијаграма 15 табеле за административно особље треба да буду заједно, јер те инфромације које су садржане у свакој табели, могу да спадају и у категорије других табела и често се претражују заједно.

5.0. ЗАКЉУЧАК

На основу претходних поглавља, и детаљно размотрене теме, може се закључити да је UML настао обједињавањем већег броја објектно оријентисаних графичких језика за моделовање. Имајући у виду да је настао обједињавањем, веома брзо се проширио и постао један од основних језика за моделовање. Основни елементи UML-а су ствари, везе и дијаграми. На дијаграме је посебан акценат стављен и постоје дијаграми случајева, дијаграми класа, дијаграми секвенци, дијаграми сарадње и промене стања, дијаграми компоненти и дијаграми развоја.

У трећем делу овог рада је разматрана могућност коришћења UML-а за моделовање база података, и сагледавајући све чињенице, закључено је да је веома погодно користити UML за моделовање база података, пре свега због широког спектра људи који су укључени у целокупни систем. Велики број људи подразумева и опширност база података, а самим тим и сложеност информација и података, па као такви морају бити читљиви и лаки за разумевање, како пројектанту, тако и корисницима, па је UML идеалан за креирање и моделовање база података. Моделовање база података се изводи користећи стандардне елементе за дизајнирање, попут шема, табела, погледа, колона, кључева, индекса и веза.

У раду је такође размотрено и коришћење Microsoft Visio софтвера који се веома добро показао за моделовање база података, па је целокупно време потребно за креирање исте, сведено на минимум, што је и један од основних циљева UML-а.

И најзад одрађен је студијски пример коришћења UML-а за базе података, и приказ његовог пребацивања у физички модел.

Закључно мишљење је да је UML најмоћнији софтвер на светском нивоу за графичко пројектовање, а нарочито се добро показао за дизајнирање и моделовање база података.

6.0. ЛИТЕРАТУРА

[1] UML-moivacija, vrste I primeri, Ognjen Kocić, Sanja Mijalković, Predrag Vujić, Matematički fakultet u Beogradu, 11000 Beograd, Srbija

[2] http://www.vps.ns.ac.rs/Materijal/mat14829.pdf (pregledano 28.06.2015)

[3] http://www.ef.uns.ac.rs/Download/razvoj_is/2009-05-08_test2-deo1_su.pdf (pregledano 28.06.2015)

[4] http://gopcevic.5gbfree.com/spiro/programiranje/100-UMLIntroduction.pdf (pregledano 29.06.2015)

[5] Veljović, V, A., (2005), Praktikum iz projektovanja informacionih sistema, Fakultet za poslovne studije, Beograd, str. 13

[6] Fowler, M., Scott, K., (2004), *UML ukratko,* Mikro knjiga, str. 112.

[7] http://rti.etf.bg.ac.rs/rti/ir4ps/predavanja/UML/06%20Dijagrami%20Interakcije.pdf

[8] Tomašević, V., (2012), *Razvoj aplikativnog softvera,* Univerzitet Sungidunum, Beograd, str. 89.

[9] http://www.ef.uns.ac.rs/Download/razvoj_is/2009-05-08_test2-deo1_su.pdf]

[10] http://rti.etf.bg.ac.rs/rti/ir4ps/predavanja/UML/06%20Dijagrami%20Interakcije.pdf (pregledano 28.06.2015)

[11] http://www.link-elearning.com/site/kursevi/lekcija/5496 (pregledano 30.06.2015)

[12] IBM Official Library, preuzeto 17. 07. 2015.

[13] Blaha M., Premerlani W.: **Object-Oriented Modeling and Design for Database Applications**, Prentice'Hall, Inc., New Jersey, 1998.

[14] Вељовић А.: **Основе објектног моделирања – UML**, Компјутер библиотека, Чачак, 2002.

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

[16] Лазаревић Б.: **Базе података**, ФОН Београд, Београд 2003.

[17] Полишчук Ј.: **Пројектовање информационих система**, Електротехнички факултет, Подгорица,2007.

[18] Rainer K., Turban E.: **Introduction to Information Systems**, John Wiley & Sons, Inc., 2009