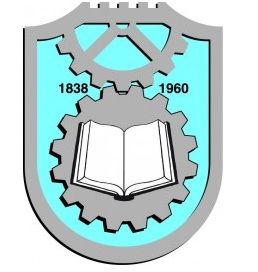
Универзитет у Крагујевцу

Факултет инжењерских наука



**ПРОЈЕКТНИ ЗАДАТАК**

**Предмет:**Софтверски инжењеринг

**Тема:**Симулација игре Х – О (икс-окс)

**Професор**: **Студент:**

Ненад Филиповић Новак Пајовић 625/2017

Тијана Шуштершич

Садржај

[**Опис пројекта** 3](#_Toc49893418)

[*Табела* 4](#_Toc49893419)

[*Метода за пролаз кроз табелу* 4](#_Toc49893420)

[*Методе за слободну позицију знакова* 6](#_Toc49893421)

[*Минимакс алгоритам* 7](#_Toc49893422)

[*Учестале петље које проверавају крајњи исход* 10](#_Toc49893423)

[**UML дијаграми** 11](#_Toc49893424)

[*USE Case Diagram* 11](#_Toc49893425)

[*Дијаграм секвенци* 12](#_Toc49893426)

[*Дијаграм активности* 13](#_Toc49893427)

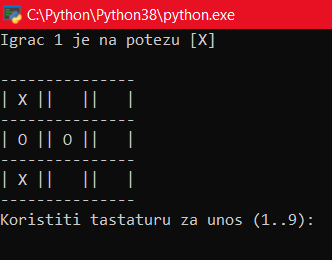
[*Дијаграм стања* 14](#_Toc49893428)

[*Дијаграм класа* 15](#_Toc49893429)

[***Литература*** 16](#_Toc49893430)

# Опис пројекта

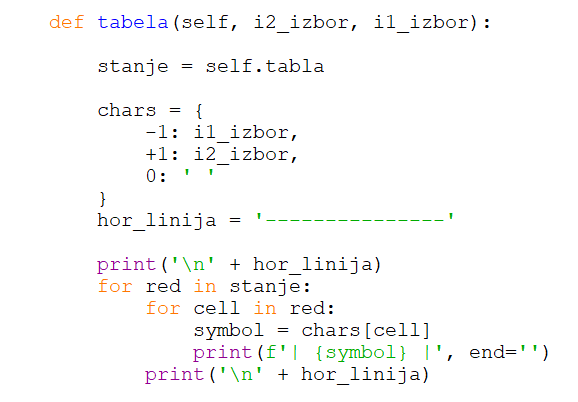
Пројекат представља типичан облик игре ИКС – ОКС који нуди две могућности: игру два играча и игру једног играча против рачунара.  
Табела је представљена матрицом облика 3x3 и поента игре је спојити три иста знака хоризонтално, вертикално или дијагонално. Уколико нико од играча не сакупи три иста знака, игра је нерешена. Корисник, при одабиру игре против другог корисника, има могућност да кликом на одређено (слободно) поље стави дати знак. Уколико корисник први игра, знак је Х (икс), односно О (окс) за другог корисника. Игра се наизменично док неко од играча не сакупи три знака или се не попуне сва поља. При одабиру игре против рачунара, принцип је исти при чему рачунар игра уместо другог корисника на основу претходно одиграних потеза првог корисника. Пројекат је рађен у програмском језику *Python.*



*Слика1. Изглед апликације након пар одиграних потеза*

## *Табела*

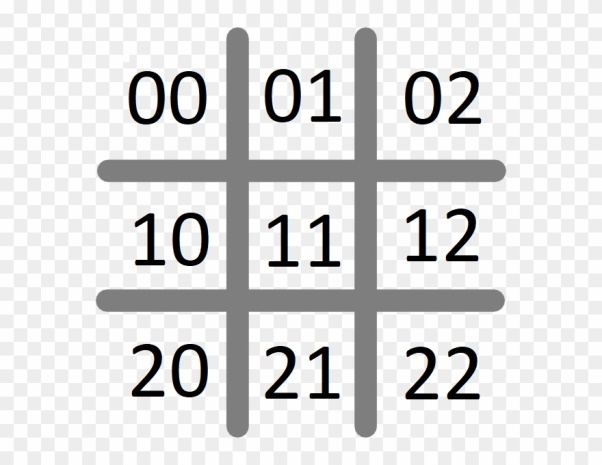
Уз помоћ методе *tabela* формира се табела која се исцртава на почетку и након сваког одиграног потеза. Табела је сачињена од девет поља која се на стандардном излазу приказују уз помоћ вертикалних и хоризонталних цртица(|, -). Коришћењем *for* петље, табела се након сваког потеза штампа на конзоли.



*Слика 2. Табела*

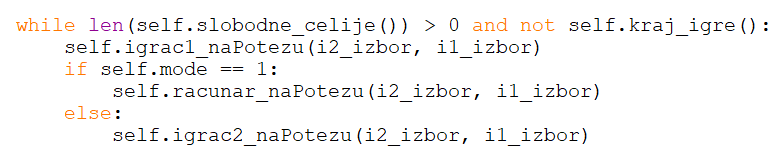
## *Метода за пролаз кроз табелу*

Матрица 3х3 представљена је индексима x и y који имају вредности 0, 1 и 2, па тако свака комбинација ова два индекса представља одређену позицију поља у матрици (слика 3).



*Слика 3. Матрица(x, y) облика 3х3*

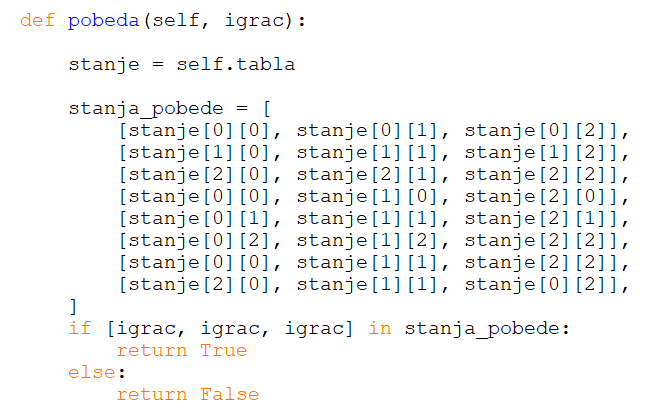
Треба обезбедити да се игра наизменично и то се ради помоћу следеће петље:



*Слика 4.*

Све док важи дати услов, тј. да има слободних поља и да игра није готова, играчи наизменично постављају знакове на слободна поља.

Играч побеђује када споји три знака узастопно по хоризонталама, вертикалама или дијагоналама, што се проверава метода *победа* на следећи начин:

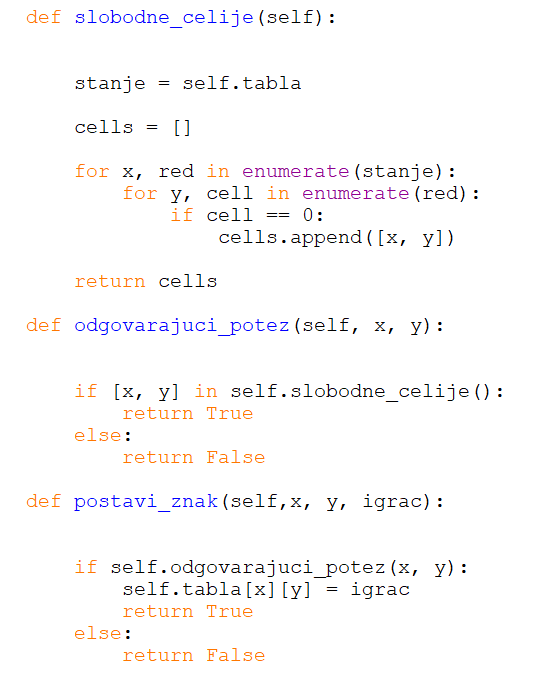


*Слика 5. Метода победа*

Уз помоћ прве три линије кода се резервишу места у реду, у следећа три колоне и у последња два дијагонале. Ако играч (рачунар) своја три знака постави на три узастопна резервисана поља, он је победио.

## *Методе за слободну позицију знакова*

Неопходно је обезбедити да се знак поставља само на слободну позицију, па се зато уводе следеће методе: *slobodne\_celijе, odgovarajuci\_potez* и *postavi\_znak(наведене методе представљене у коду, слика 6).* Метода *slobodne\_celije* враћа низ слободних ћелија у табели са њиховим координатама х и y. Двема for петљама пролази се кроз табелу и проверавају се слободне позиције које се пакују у низ. Метода *odgovarajuci\_potez* проверава да ли је одигран одговарајући потез тј. да ли је изабрано празно поље. Метода *postavi\_znak* позива претходну методу и има два излаза. Ако је претходна метода вратила True, онда ова метода на изабрану позицију смешта знак играча који је на потезу и враћа True, у супротном враћа False.



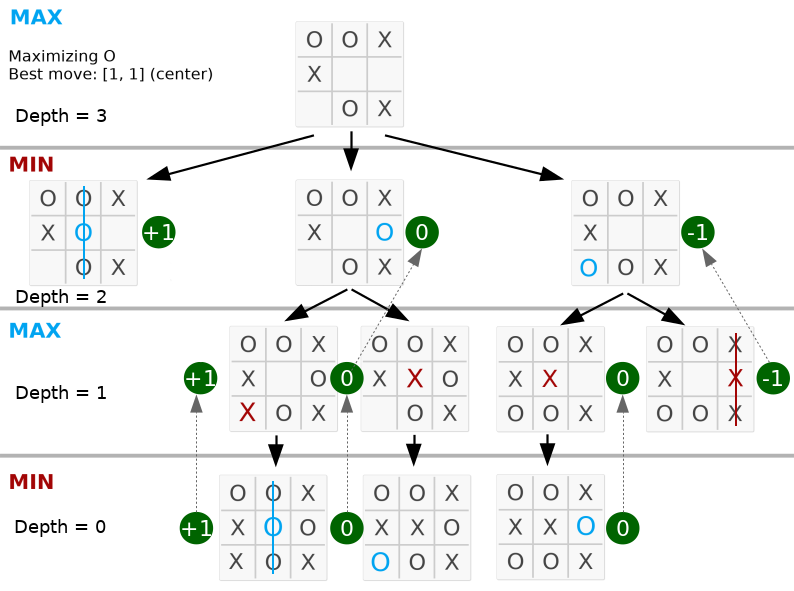
*Слика 6. Методе*

## *Минимакс алгоритам*

**Минимакс** је правило одлука којe се користи у теорији одлучивања, теорији игара, статистици и филозофији за минимизирање могућих губитака у најгорим случајевима. Првобитно је формулисан за теорију нулте-суме два играча, који покрива оба случаја где играчи узимају алтернативне потезе и оне где чине симултане потезе, такође је проширена на више комплексних игара и нa опште одлучивање у присуству неизвесности.

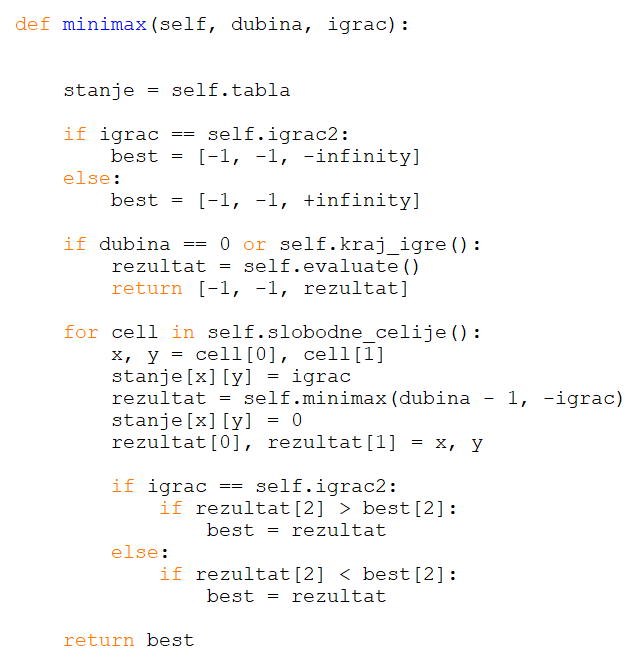
Сврха алгоритма у примеру игре ХО је да први играч (Х) максимизује свој резултат и одигра најбољи потез и тако минимизује резултат свог противника. Уколико, играч не може да максимизује свој резултат, тражи следеће најбоље решење за њега(у случају игре ХО, то би био нерешен резултат). Ако играч не успе да максимизује свој резултат или бар не обезбеди нерешен резултат, онда даје шансу противнику да он максимизује свој резултат.

На следећем примеру (слика 7) приказана је ситуација након пар одиграних потеза, окс има шансу да максимизује свој резултат (+1), минимизује свој резултат(-1) и обезбеди себи нерешен резултат(0) или можда још једну шансу за победу. У случају максимизације окс доноси победу постављањем знака на позицију 4 и односи победу. Уколико окс постави знак на позицију 2, тада минимизује своју шансу за победу и максимизује противников резултат(дакле, има шансу или да изгуби или да буде нерешено). У последњем случају, окс постављањем знака на позицију 7, обезбеђује нерешен резултат у најгорем случају, а у најбољем победу.



*Слика 7. Пример Минимакс алгоритма*

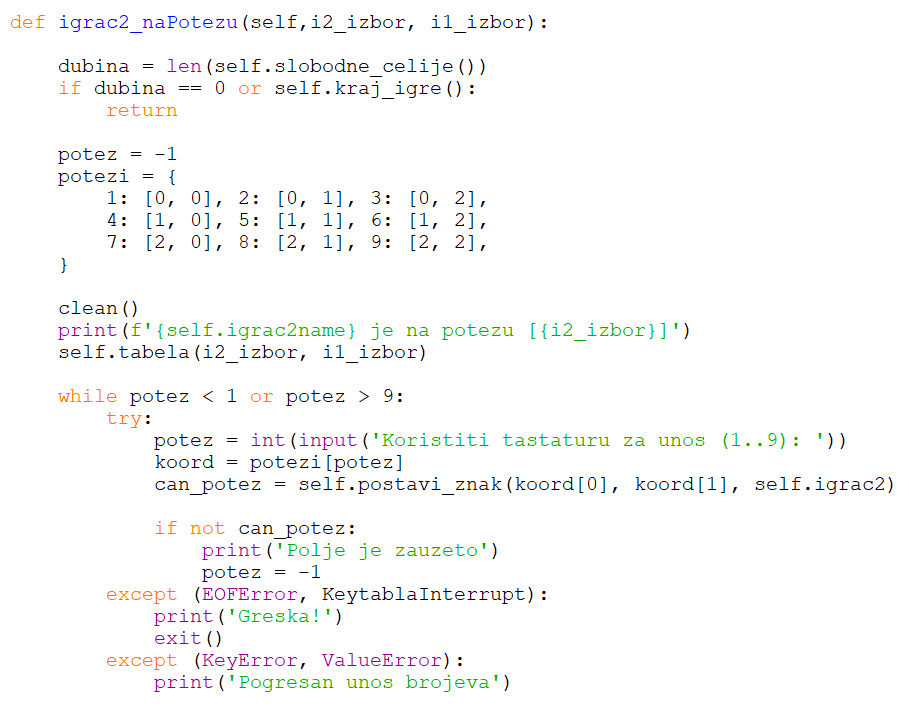
Минимакс алгоритам се користи у моду игре корисника против рачунара и има следећи приказ:



*Слика 8. Минимакс алгоритам приказан у коду*

Минимакс алгоритам након сваког одиграног потеза пролази кроз свако слободно поље на табели и проналази најбољи резултат за њега. Рачунар функционише по принципу спречавања противника да победи, самим тим минимизује његов резултат. Проласком кроз табелу и наиласком на могућност противника да победи (односно споји три знака у низу), рачунар га спречава стављањем његовог знака на то место.   
  
*Играч против играча*

За опцију игре играча против играча, корисник на самом почетку може покренути тај мод уносом броја “2” у конзолу. На идентичан начин се врши уношење знакова за првог и за другог играча, дозвољавајући коришћење само бројева од један до девет (због девет поља у табели). Овако то изгледа у коду (слика 9):

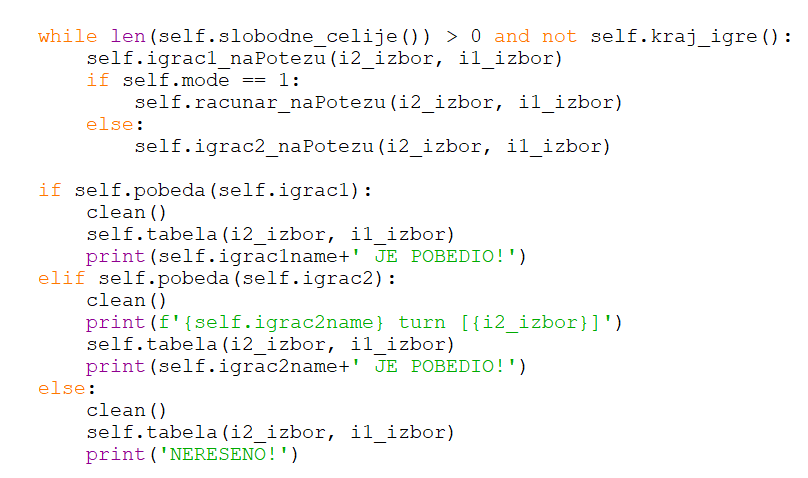


*Слика 9.*

Уколико изаберемо мод два играча, играчи произвољно бирају поља на која смештају своје знакове. Програм користи исте методе које су претходно наведене и проверава да ли се три узастопна знака налазе по хоризонталама, вертикалама, дијагоналама или су сва поља попуњена без победника.

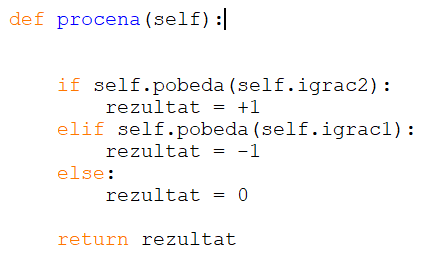
## *Учестале петље које проверавају крајњи исход*

У склопу методе *започниИгру* налазе се петље које на крају свакe партије проверавају ко је од играча победник односно да ли је партија нерешена. У коду то изгледа овако:



*Слика 10. Метода започниИгру*

Метода *процена* враћа резултат -1 ако победи први играч, тј. +1 уколико победи други играч(рачунар), што је већ споменуто у дефиницији минимакс алгоритма.



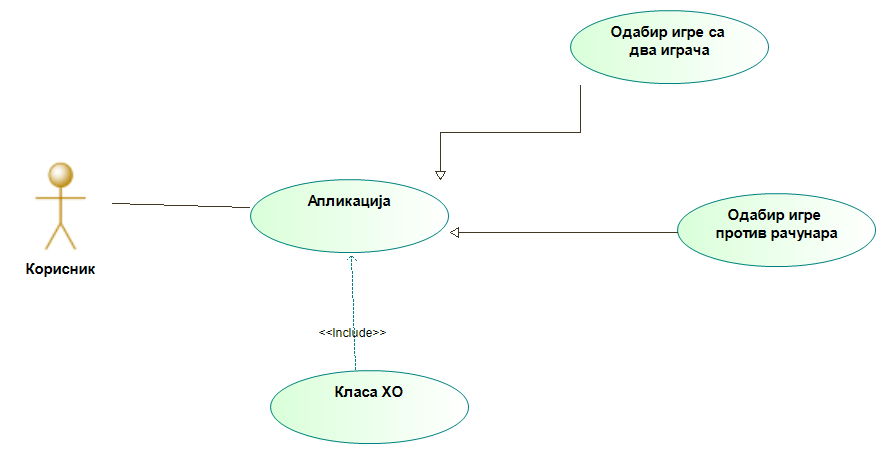
*Слика 11. Метода процена*

# UML дијаграми

## *USE Case Diagram*

* Дијаграм случајева коришћења приказује скуп случајева коришћења и актера
* Oмогућавају крајњим корисницима да разумеју систем
* Поглед корисника на функционисање система(шта систем ради, а не како систем функционише)

Дијаграм случајева за дати пројекат, приказан је следећом сликом:

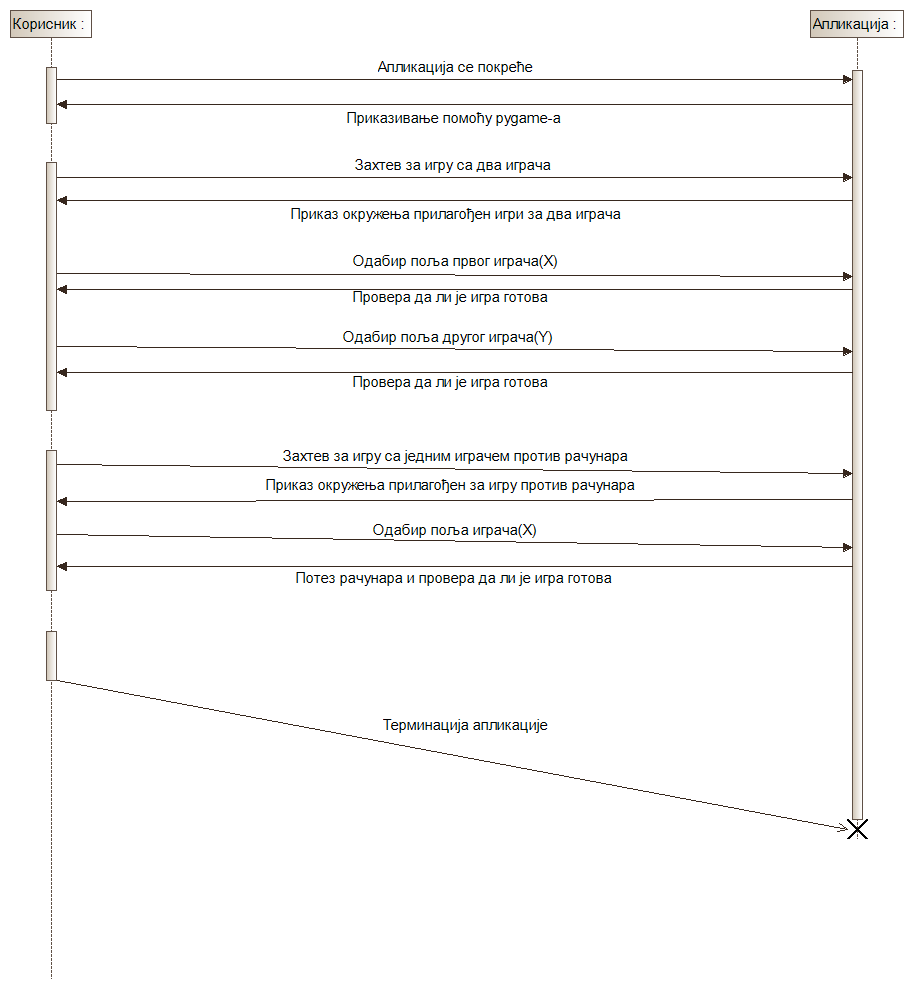


*Слика 12. Use Case Diagram*

## *Дијаграм секвенци*

* Користе се да специфицирају интеракцију
* Моделирају конкретне сценарије
* Описујући комуникацијске секвенце на различитим нивоима детаља

Дијаграм секвенци за дати пројекат, приказан је следећом сликом:



*Слика 13. Дијаграм секвенци*

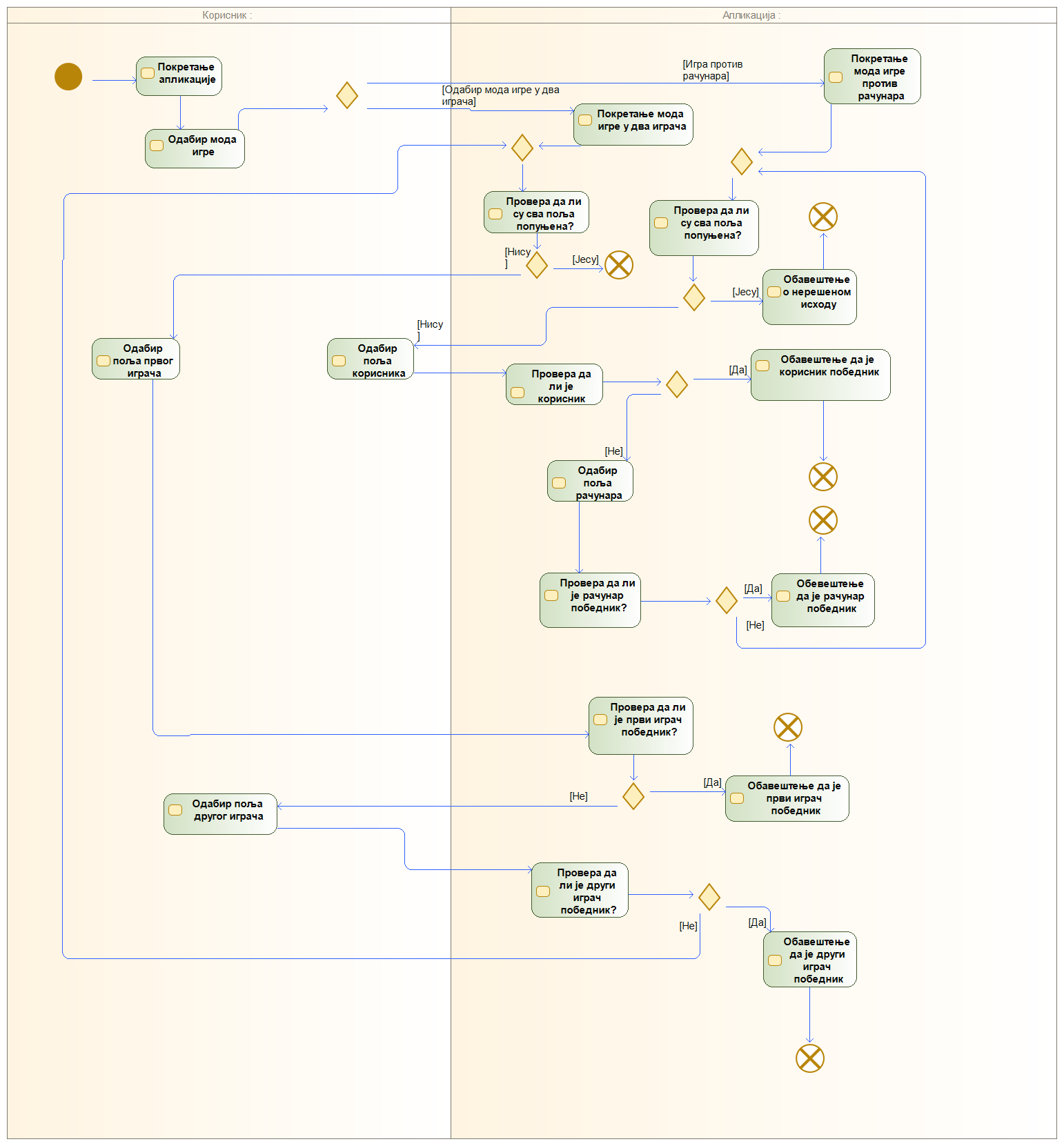
## *Дијаграм активности*

Дијаграми активности су намењени моделирању динамичких аспеката(понашања) система.

Дијаграм активности приказује:

* Ток активности коју извршавају објекти
* Евентуално и ток објеката између корака активности

Дијаграм активности за дати пројекат, приказан је следећом сликом:



*Слика 14. Дијаграм активности*

## *Дијаграм стања*

Аутомат стања/Машина стања(state machine)

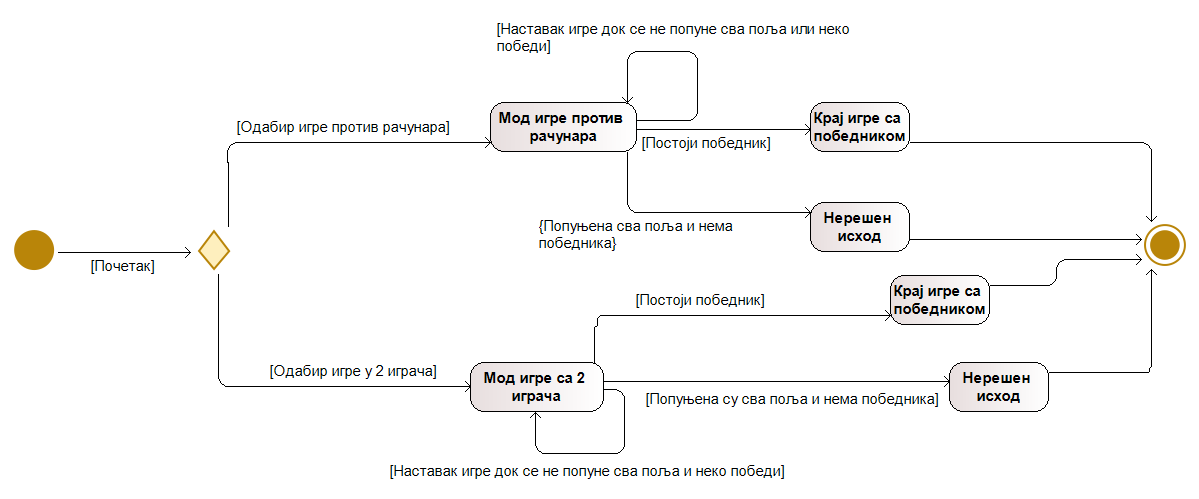
* Понашање које специфицира секвенце стања кроз која пролази
* Моделира **понашање** неког ентитета или **протокол** интеракције

Дијаграм стања је граф који приказује аутомат стања:

* Чворови су стања
* Гране су прелази

Дијаграми стања се фокусирају на ток активности.

Дијаграм стања за дати пројекат, приказан је следећом сликом:



*Слика 15. Дијаграм стања*

## *Дијаграм класа*

# *Литература*

1. <http://moodle.fink.rs/course/view.php?id=978>
2. <https://en.wikipedia.org/wiki/Unified_Modeling_Language>
3. <https://sr.wikipedia.org/sr-ec/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BA%D1%81_(%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%B0%D0%BC)>
4. <https://docs.python.org/3/tutorial/>
5. <https://www.modelio.org/documentation-menu/tutorials.html>