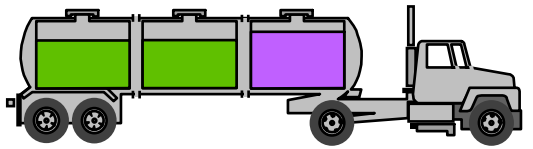
Objektni pristup razvoja informacionog sistema

Osnovi koncepti objektne orjentacije

- Bilo koji sistem se može posmatrati kao skup međusobno povezanih objekata
- Pod objektima u nekom sistemu se podrazumevaju fizički objekti, koncepti, abstrakcije, bilo šta što ima jasne granice i jasno značenje, što se jasno razlikuje od drugih objekata u sistemu
- Pod objektom se podrazumeva entitet koji je sposoban da čuva svoja stanja i koji stavlja na raspolaganje okolini skup operacija preko kojih se ta stanja prikazuju ili menjaju
- U strogo objektnim pristupima jedini način da se pristupi stanjima objekta ili da se ona promene je preko neke, iz skupa definisanih, operacija
- Jedini vidljivi deo objekta su operacje i to ne način na koji su implementirane, već samo njihovi efekti (specifikacija)

Šta je objekat ?

- Neformalno, objekat pretstavlja neki entitet bilo fizički, konceptualni ili logički(software)
  - Fizički entitet

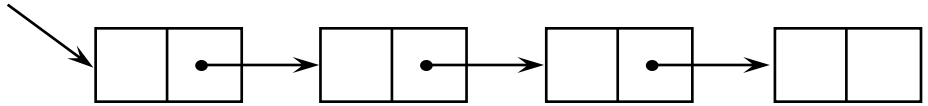


Kamion

  - Konceptualni entitet

Hemijski proces

  - Softverski entitet



Jednostruko spregnuta lista

Formalna definicija objekta

- Objekat pretstavlja koncept, apstrakciju ili stvar sa jasnim granicama i značenjem sa aspekta neke primene
- Objekat je nešto što poseduje:
  - Stanje
  - Ponašanje
  - Identitet

Kreiranje apstrakcije objekta

- Treba razlikovati objekte u realnom svetu i softversku predstavu objekta
- Apstrakcija podrazumeva eliminisanje detalja da bi se ilustrovale glavne tačke nekog koncepta
- Apstrakcija – predstava objekta iz realnog sveta
- Apstrakcije nam omogučavaju rukovanje složenošću koncentracijom na esencijalne karakteristike po kojima se jedan entitet razlikuju od drugih

Stanje objekta

- Stanje objekta se pretstavlja vrednostima njegovih osobina.
- Pod osobinama se podrazumevaju atributi objekta i njegove veze sa drugim objektima u sistemu
- Osobine objekta se menjaju u vremenu
- Ponašanje objekta se opisuje preko skupa operacija koje on izvršava
- Svaka operacija može da ima listu ulaznih i izalaznih parametara definisanih tipova, a može i da vrati tipiziran rezultat

Tip objekta

- Objekti se grupišu u “tipove“
- Svi objekti istog tipa imaju zajednički skup osobina (atributa)
- Svi objekti istog tipa imaju zajedničko ponašanje (isti skup operacija)

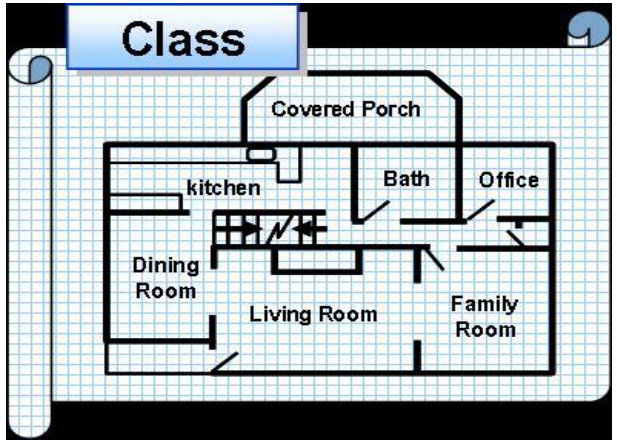
Šta je klasa?

- Klasa je opis grupe objekata koji poseduju skup zajedničkih: svojstava (atributa), ponašanja (operacije/metode)
  - Objekat je instanca(primerak, pojavljivanje) neke klase
- Klasa pretstavlja apstrakciju jer:
  - Naglašava relevantne karakteristike
  - Potiskuje sve ostale karakteristike

Odnos klase i objekata

- Klasa :
  - To je model koji opisuje kako kreirati objekat
  - je kao “šematski plan(skica)”
  - Sadrži podatke i metode
- Objekti:
  - Objekat je predstava nekog entiteta iz realnog sveta
  - Instance klase
  - Može biti više objekata (instanci) klase

Primer klase i objekata



Objekti

Uvod u UML

Model i modelovanje

- Razvoj informacionog sistema se u velikoj meri svodi na nalažanje odgovarajućeg modela realnog sistema
- Sistem se definiše kao skup objekata (entiteta) i njihovih međusobnih veza
- Model je "subjektivni odraz objektivne stvarnosti“
- Može da postoji više različitih modela istog sistema, sa istog ili različitih aspekata
- Model je neko pojednostavljenje realnosti
- Modeli se izgrađuju da bi se bolje razumeo realni sistem

Modelovanje

- Modelovanje je način da se savlada složenost nekog sistema
- Modelovanje je opšti pristup u svim inženjerskim disciplinama
- U svakoj oblasti postoji, više, alata za modelovanje sistema
- UML(Unified Modelling Language) je alat za modelovanje softverskih sistema

Šta je UML

- UML – Opšti vizuelni jezik za modelovanje softverskih sistema
- Svrha UML-a:
- **Vizuelizacija** – grafički prikaz strukture i ponašanja sistema
- **Specificiranje** – precizno i jednoznačno definisanje modela
- **Konstruisanje** – osnova za razvoj softverskog sistema (nije programski jezik, ali može biti povezan sa generisanjem koda)
- **Dokumentovanje** – standardizovani simboli za kreiranje tehničke dokumentacije

Prednosti korišćenja UML-a

- Softverski sistem se profesionalno dizajnira i dokumentuje pre nego što se počne sa kodovanjem
- Kada je potrebno nadograđivati sistem mnogo je lakše raditi sa sistem koji ima UML dokumentaciju.
- Troši se manje vremena na ponovno učenje sistema.
- Troškovi održavanja sistema su manji

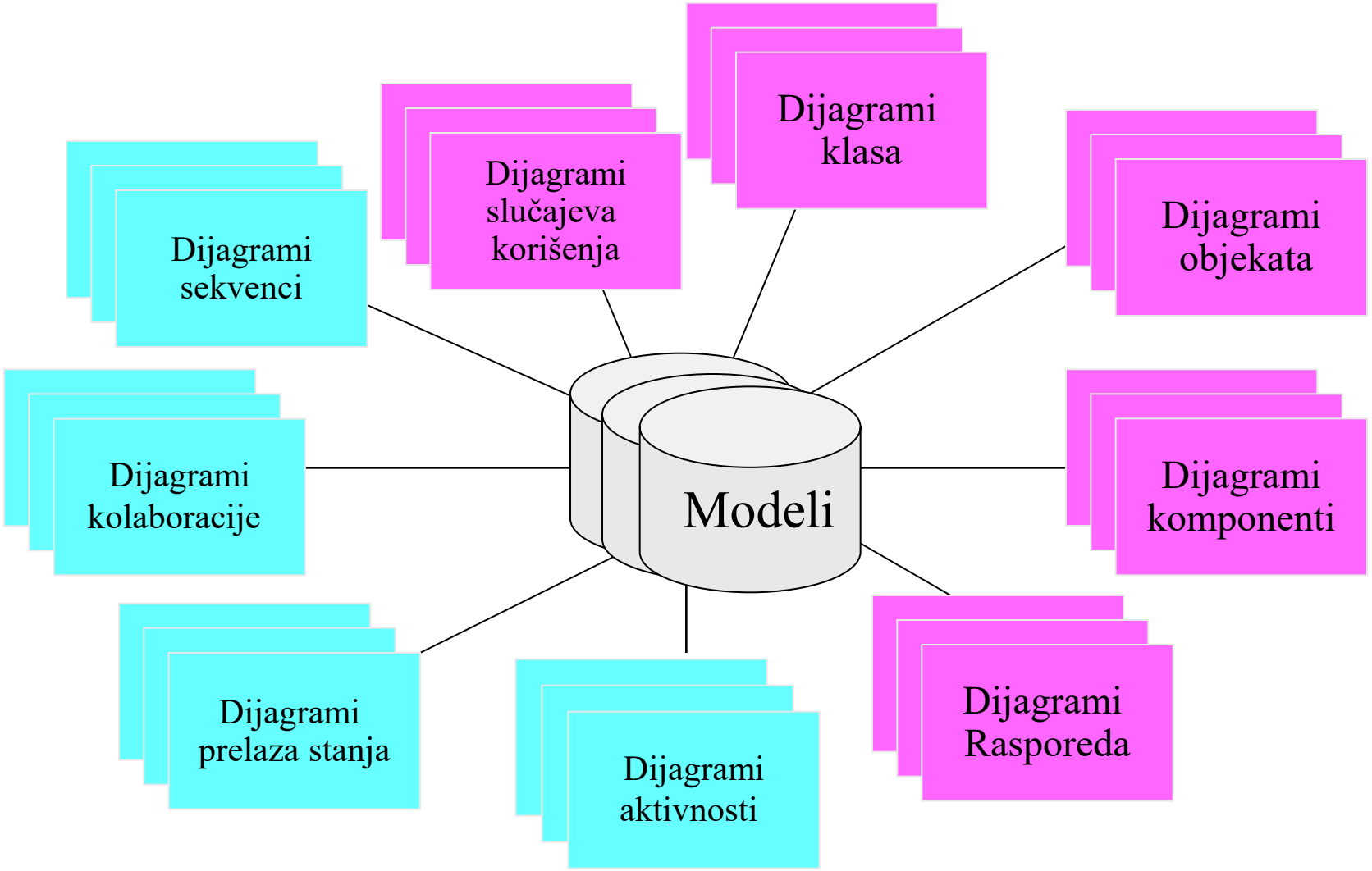
Prednosti korišćenja UML-a

- Omogućava efikasniji timski rad
- Neko se lakše i brže uključuje u projekat ukoliko postoji UML dokumentacija
- Ukupni troškovi sistema su manji
- Dobija se pouzdaniji i efikasniji softver, kao i softver koji se može ponovo iskoristiti
- Važne odluke se donose još u fazi dizajna sistema pre nego što se napiše loš kod

Polazni koncepti UML-a

- **Objekat** (model entiteta realnog sveta)
  - Ima svoj identifikator
  - Stanje
  - Ponašanje
- **Klasa** (model skupa entiteta realnog sveta koji imaju neke zajedničke osobine)
  - Skup atributa
  - Skup operacija
- **Veza** (model odnosa između klasa ili objekata)
- **Poruka** (oblik moguće interakcije između objekata)

Modeli i dijagrami



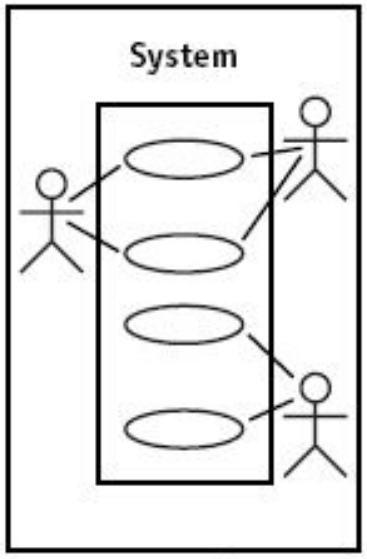
Organizacija UML dijagrama korišćenjem pogleda

- UML dijagrami se organizuju poglede:
- **Funkcionalni pogled** – prikazuje šta sistem radi iz ugla korisnika i njihovih potreba (npr. Use Case Diagram, Activity Diagram)
- **Statički pogled** – opisuje statičke elemente sistema kao što su klase, paketi, objekti i njihovi odnosi (npr. Class Diagram, Component Diagram)
- **Dinamički pogled** – fokusira se na interakcije i tokove aktivnosti tokom vremena (npr. Sequence Diagram, State Machine Diagram)

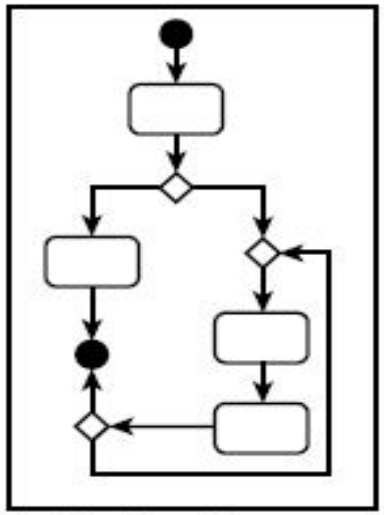
Funkcionalni pogled

- Uključuje dijagrame slučajeva korišćenja (**use case diagram**) i dijagrame aktivnosti (Activity diagram.)
- **Use Case** diagram opisuje karakteristike koje sistem treba da obezbedi korisniku tj. opisuje ponašanje sistema sa stanovišta korisnika
- **Dijagram aktivnosti** prikazuje tok procesa, uključujući redosled, grananja i paralelne ativnosti i sličan je dijagramu toka (flowchart)

Dijagrami funkcionalnog pogleda



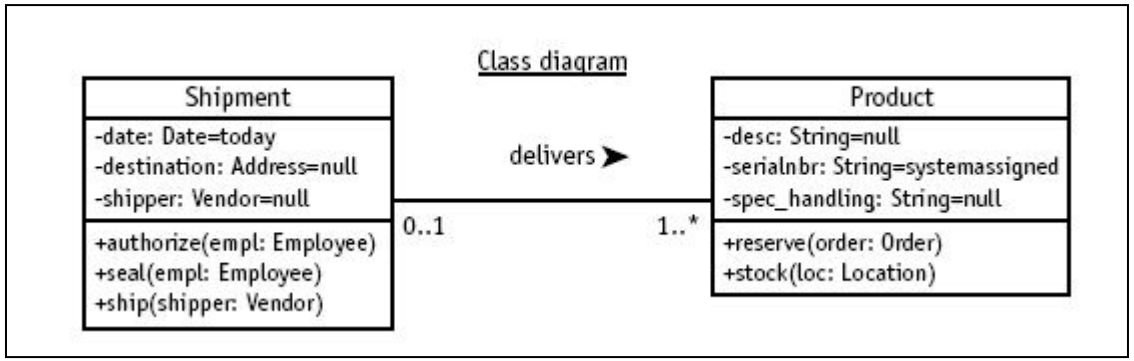
Use Case Diagram

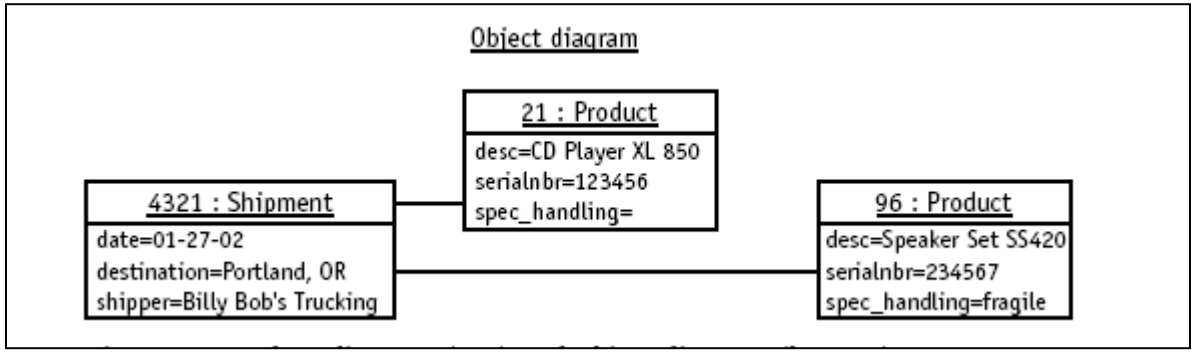

Activity Diagram

Statički pogled

- Predstavlja skicu elemenata sistema ali ne govori ništa o ponašanju sistema
- **Klasni dijagrami** za predstavu klasa
- **Objektni dijagrami** za predstavu objekata klase – imaju specifične vrednosti atributa i ponašanje. Koriste se za razumevanje klasnih dijagrama ili za njihovo testiranje
- Statički dijagrami obezbeđuju predstave sistema gledano iz ugla onog koji taj sistem razvija

Dijagrami statičkog pogleda

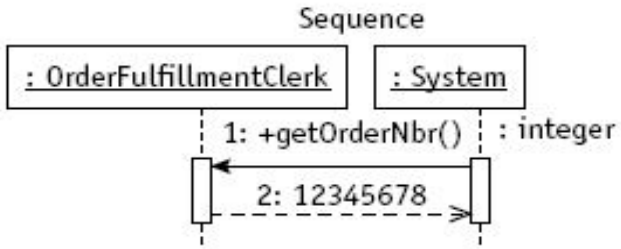




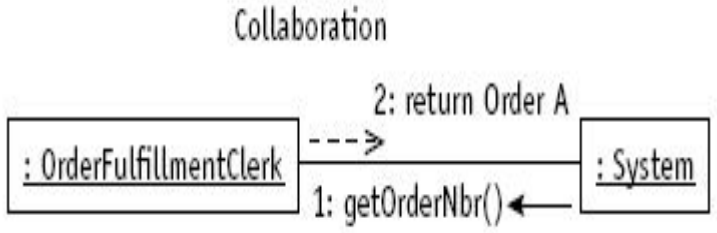
Dinamički pogled

- **Dinamički** pogled uključuje dijagrame koji objašnjavaju interakciju objekata
- **Sekvencijalni i dijagrami saradnje** se nazivaju zajedničkim imenom interakcioni dijagrami
- **Statechart** dijagrami pokazuju kako i zašto se objekti menjaju pod uticajem okoline

Dijagrami dinamičkog pogleda



Dijagrami sekvenci- vremenski redosled poruka



Kolaboracioni dijagrami,
prikazuje se ko sa kim komunicira
ali ne i kada

Statechart -prikazuje promene stanja jednog objekta tokom njegovog životnog veka, kao odgovor na događaje.

Use Case dijagrami

Slučaj korišćenja (Use Case)

- Predstavlja konkretan načine korišćenja sistema
- Opisuje interakciju između aktera (korisnika) i sistema
- Fokusirase na funkcionalnosti koje sistem pruža korisniku
- Skup slučajeva korišćenja predstavlja kompletno ponašanje sistema
- Use Case se obično imenuje kombinacijom glagol + imenica
  - npr. „Kreiranje računa“, „Podizanje novca“
- Skup svih Use Case-ova daje pregled funkcionalnosti sistema

Opis slučaja korišćenja

- Svaki slučaj korišćenja treba da bude detaljno opisan
  - preporučuje se strukturirani **verbalni opis**, posebno u početnoj fazi
  - formalni opisi (dijagrami) mogu se koristiti kao dopuna
- Opis obično uključuje:
  - normalan tok događaja
  - izuzetke i alternativne tokove
- Slučaj korišćenja predstavlja skup sekvenci događaja
  - jedna sekvenca se naziva scenario
  - razlikuju se **osnovni scenario** i **alternativni scenariji**

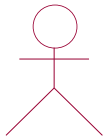
Use Case dijagram

- **Use Case** dijagram je graf sa dve vrste čvorova: čvorovi koji predstavljaju **aktere** i čvorovi koji predstavljaju **slučajeve korišćenja**
- **Use Case** dijagram grafički prikazuje**:**
  - aktere
  - slučajeve korišćenja
  - njihove međusobne odnose
- **Akter** je objekat van sistema koji predstavlja tip (vrstu) korisnika i stupa u interakciju sa informacionim sistemom
  - može biti korisnik (čovek) ili drugi sistem
- Važno je razlikovati korisnika i aktera:korisnik je konkretna osoba koja koristi sistem, dok je akter specifična uloga koju korisnik ima u komunikaciji sa sistemom

Karakteristike dijagrama slučajeva korišćenja

- Slučajevi korišćenja definišu domet sistema
- Skup svih slučajeva korišćenja predstavlja ceo sistem
- Slučajevi korišćenja su slični zahtevima ali su mnogo jasniji i fokusiraniji
- Služe kao metod za planiranje razvoja sistema i planiranje vremena potrebnog za taj razvoj
- Ne treba svaka interakcija između korisnika i sistema da predstavlja slučaj korišćenja
- Korišćenje velikog broja malih slučajeva korišćenja dovodi do velike kompleksnosti dijagrama

UML simboli u dijagramu SK

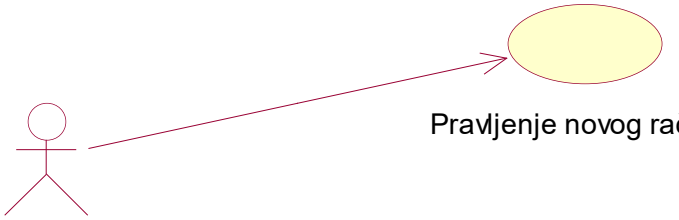


Prodavac

oznaka aktera

Pravljenje novog računa

use case



use case dijagram

Pojmovi u dijagramu SK

- **Akter** je neko ko može da inicira use case
- **Use Case** (slučaj korišćenja) je konkretna funkcionalnost koju sistem pruža korisniku
- Use case dijagrami nastaju povezivanjem aktera sa use case –om (slučaj korišćenja ili slučaj upotrebe)
- Akter inicira use case i akter je taj koji prima odgovor, stim što to ne mora biti akter koji je use case inicirao

Opis slučaja korišćenja

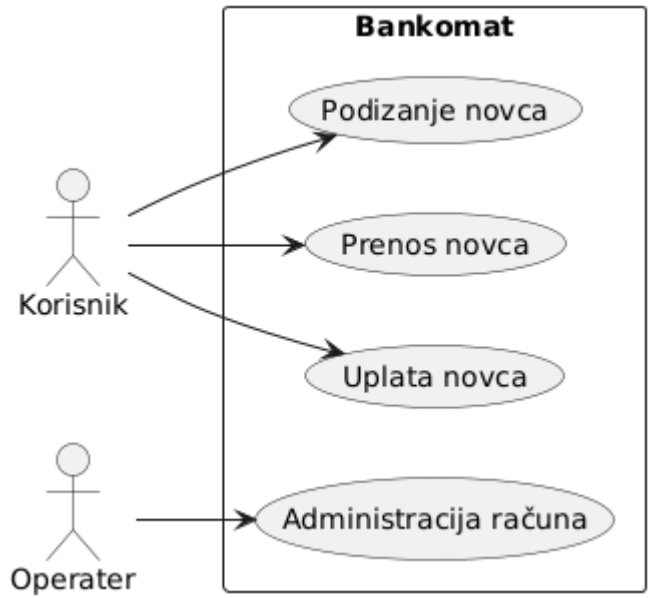
- Ime slučaja korišćenja (SK)
- Akter (SK)
- Učesnici (SK)
- Preduslovi koji moraju biti zadovoljeni da bi (SK) počeo da se izvršava
- Osnovni scenario izvršenja (SK)
- Postuslovi koji moraju biti zadovoljeni da bi SK bio uspešno izvršen
- Alternativna scenarija izvršavanja (SK)

Informacije o proizvodima (Use Case)

Use Case: Informacije o proizvodima

Element	Opis
Ime slučaja korišćenja	Informacije o proizvodima
Akter	Korisnik
Učesnici	- Korisnik (primarni akter) - Sistem za prikaz proizvoda (sekundarni učesnik)
Preduslovi	- Korisnik ima pristup aplikaciji ili veb sajtu - Sistem ima dostupne podatke o proizvodima
Osnovni scenario	1. Korisnik pristupa sekciji „Proizvodi” 2. Sistem prikazuje listu proizvoda 3. Korisnik bira proizvod 4. Sistem prikazuje detalje o proizvodu
Postuslovi	- Korisnik je uspešno pregledao informacije o proizvodu - Sistem je prikazao tačne podatke
Alternativna scenarija	**ALT1:** Proizvod nije dostupan – sistem prikazuje poruku **ALT2:** Greška u učitavanju – sistem prikazuje poruku o grešci **ALT3:** Korisnik koristi pretragu ili filtere ako ne pronađe proizvod

Primer Use Case dijagrama



Podizanje novca: osnovni scenario

- **Provera kartice**
  - Komitent ubacuje karticu
  - Sistem proverava karticu i, ako je validna, traži unos PIN-a
- **Provera PIN-a**
  - Komitent unosi PIN
  - Ako je ispravan, sistem omogućava izbor transakcije
- **Izbor transakcije**
  - Komitent bira „podizanje novca“
  - Sistem prikazuje dostupne račune
- **Unos iznosa**
  - Komitent bira račun i unosi iznos.
  - Sistem šalje zahtev banci
- **Izvršenje transakcije**
  - Sistem odobrava isplatu i priprema potvrdu
- **Kraj**
  - Sistem vraća karticu i izdaje potvrdu

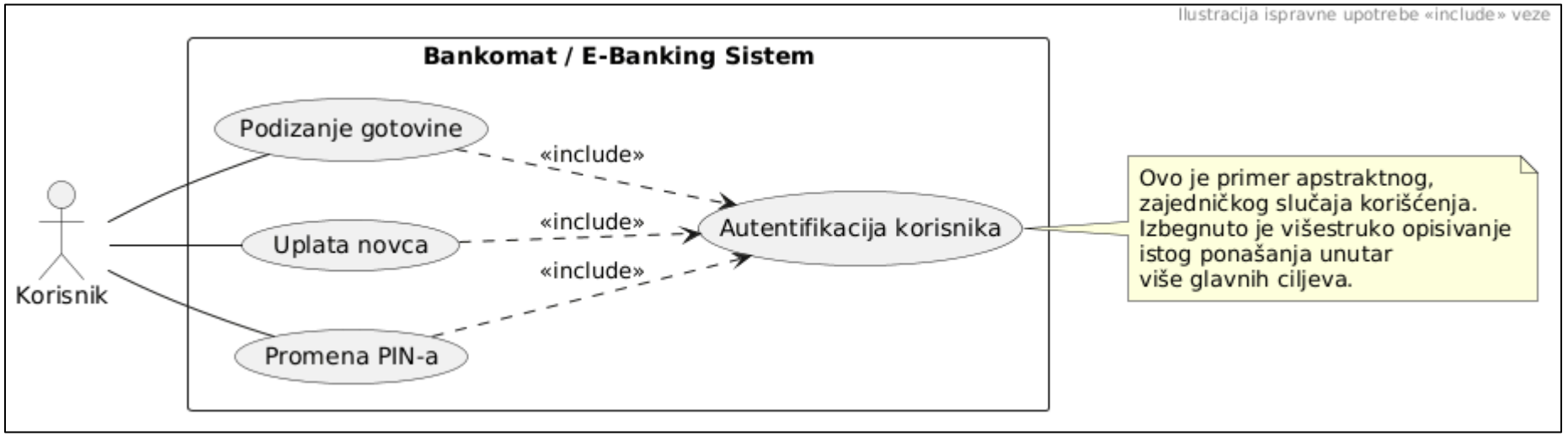
Podizanje novca: alternativni scenariji

- **Kartica nije prihvatljiva**
  - Sistem vraća karticu korisniku uz zvučni signal
- **Neispravan PIN**
  - Sistem prikazuje poruku o grešci
  - Korisniku se omogućava ponovni unos PIN-a
  - Dozvoljena su najviše tri pokušaja
  - Nakon toga sistem vraća karticu
- **Prekid transakcije**
  - Korisnik može u svakom trenutku prekinuti transakciju
  - Sistem poništava sve prethodne korake
  - Kartica se vraća korisniku

Veze u dijagramima slučajeva korišćenja

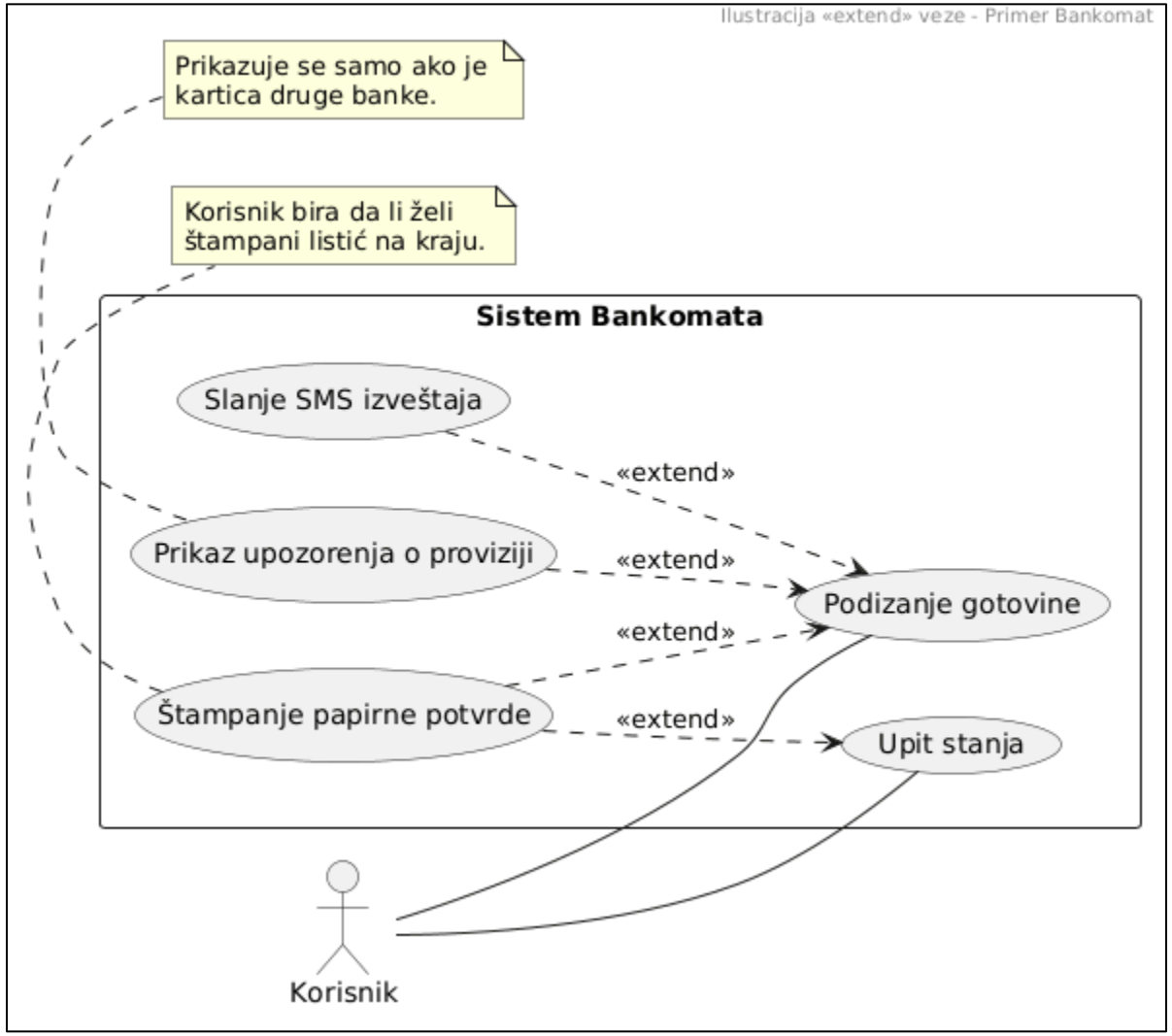
- Veza između aktera i slučaja korišćenja naziva se **asocijacija** i predstavlja komunikaciju između njih
- Asocijacija se standardno prikazuje **linijom**, bez naglašavanja smera komunikacije
- U nekim alatima može se koristiti strelica, ali ona nema posebnu semantičku ulogu u UML-u
- Generalizacija
  - odnos između opštijeg i specifičnijeg aktera ili slučaja korišćenja
- Veza **<<include>>**
  - obavezno uključivanje dodatnog ponašanja
- Veza **<<extend>>**
  - opciono proširenje osnovnog ponašanja

Ilustracije veze <<include>>



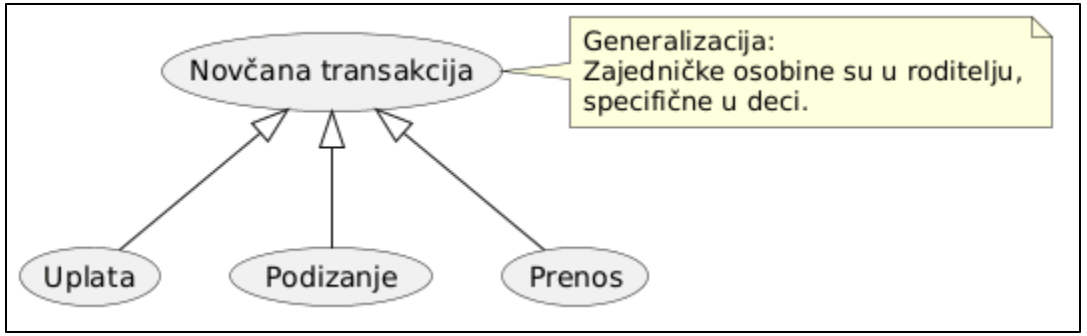
U ovom primeru, nemoguće je podići novac ili promeniti PIN bez prethodno izvršene **autentifikacije korisnika**

Primer veze <<extend>>



Veza <<extend>> označava funkcionalnost koja se izvršava samo ako se ispuni određeni uslov ili po eksplicitnom izboru korisnika.

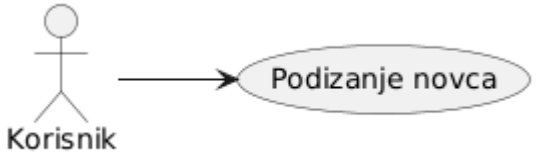
Primer generalizacije SK



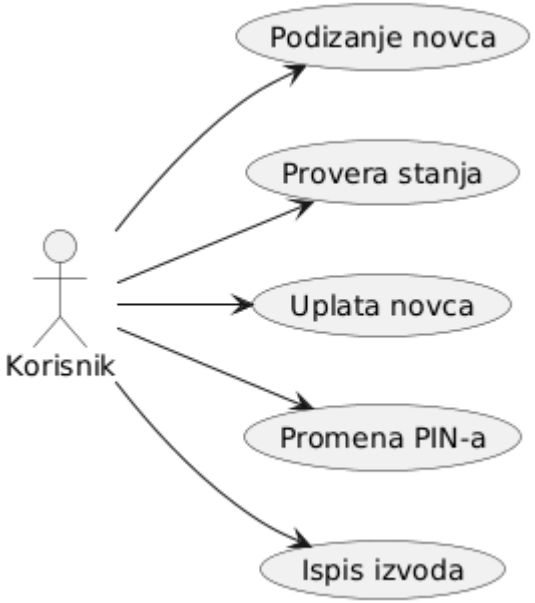
Generalizacija izdvaja zajedničke osobine u jedan opšti slučaj koji specifični slučajevi nasleđuju.

Granularnost slučajeva korišćenja

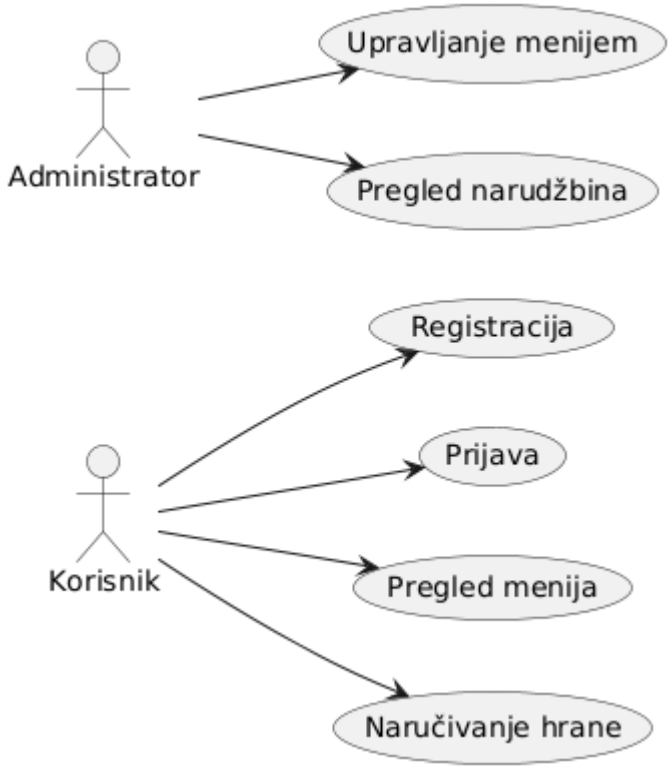
Use Case primer:



Kontra primer:
Ne treba svaka interakcija korisnika sa sistemom da predstavalja Use CAse

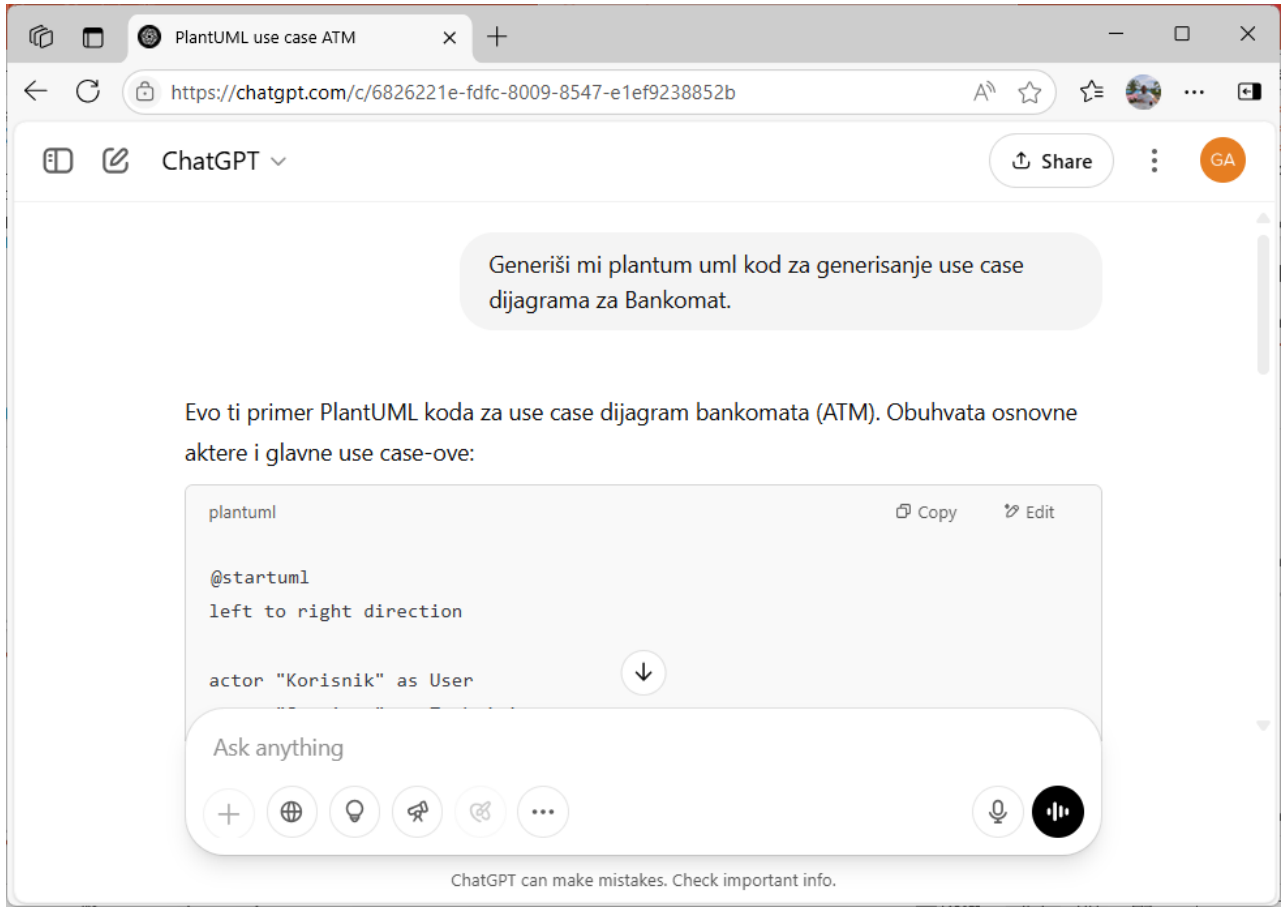


Use case dijagram – sistem za naručivanje hrane



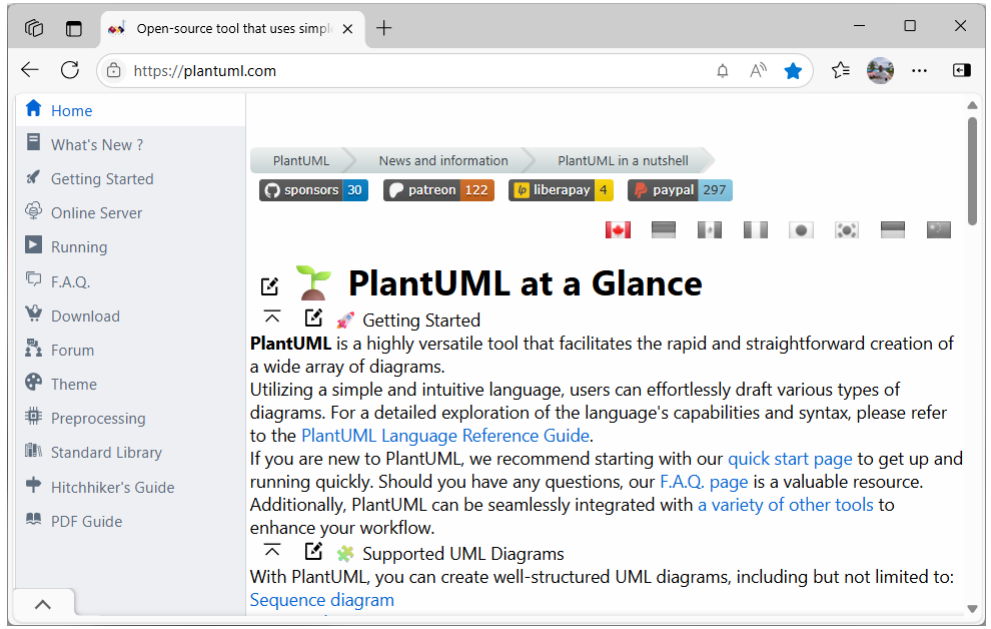
https://chatgpt.com/

Generiši mi plantum uml kod za generisanje use case dijagrama za Bankomat.



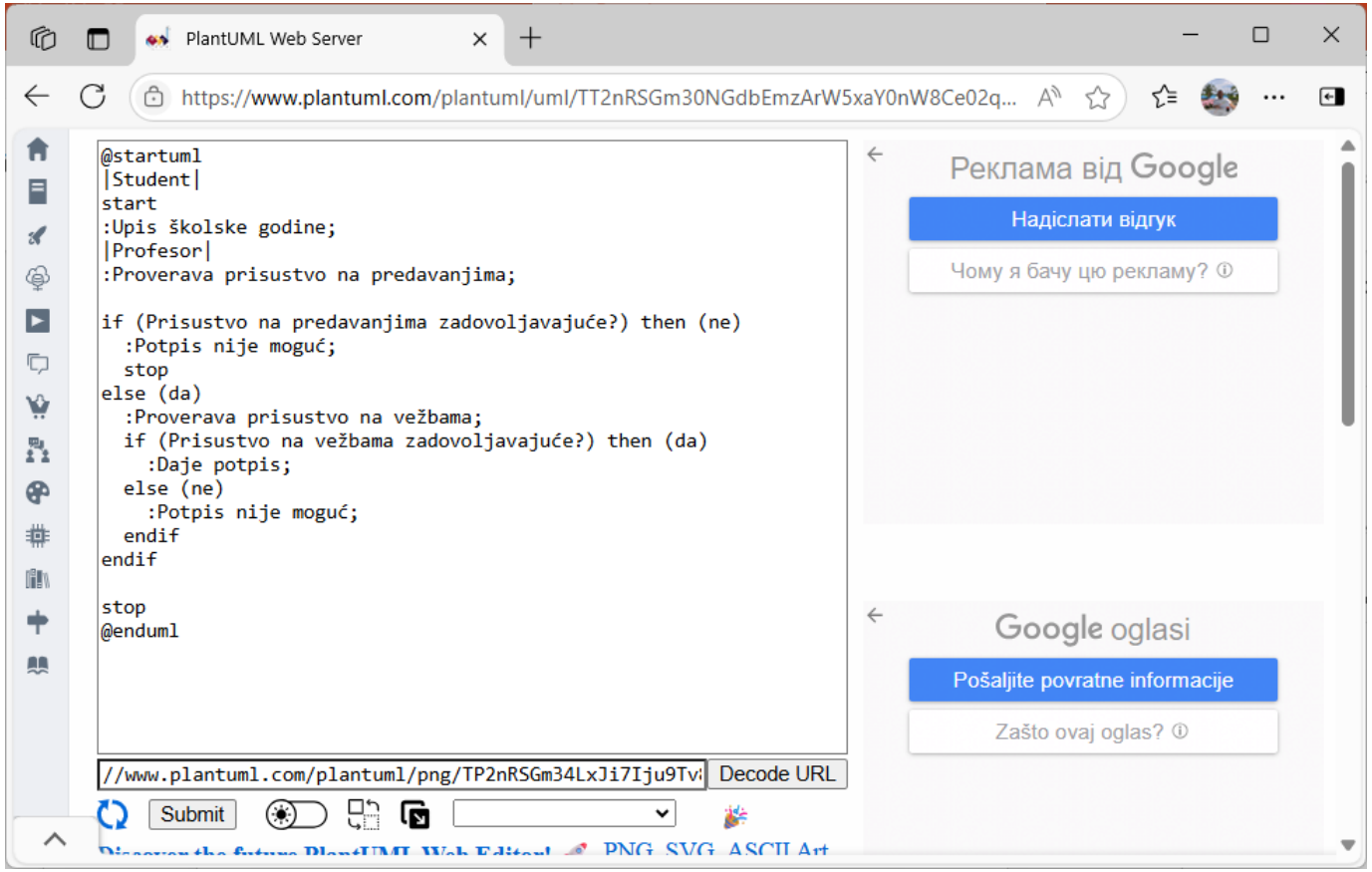
Generisanje dijagrama

https://www.plantuml.com/



Odaberi opciju **Online server**

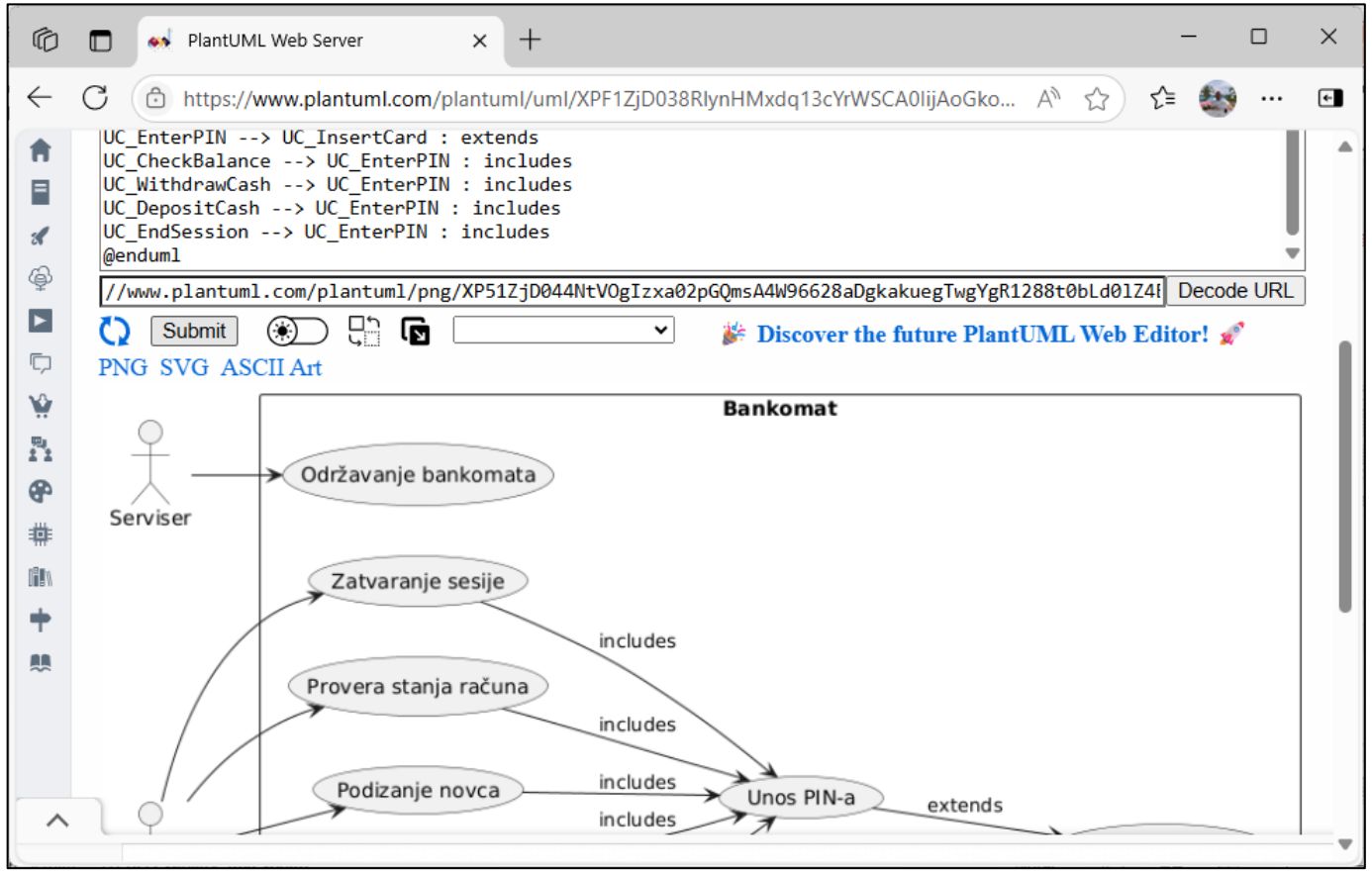
Ubacivanje plantum UM koda u editor



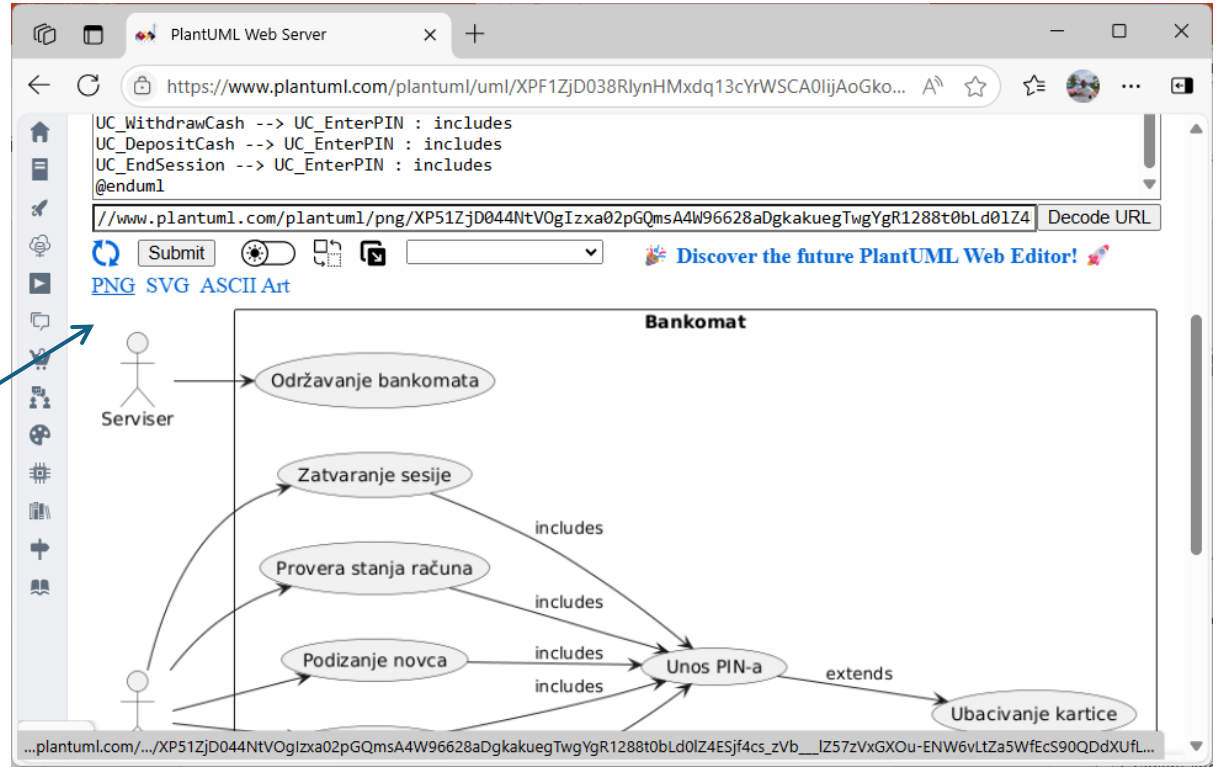
Iskopiraj kod koga je generisao Chat GPT u editor

Generisanje dijagrama

Klikni na dugme Submit

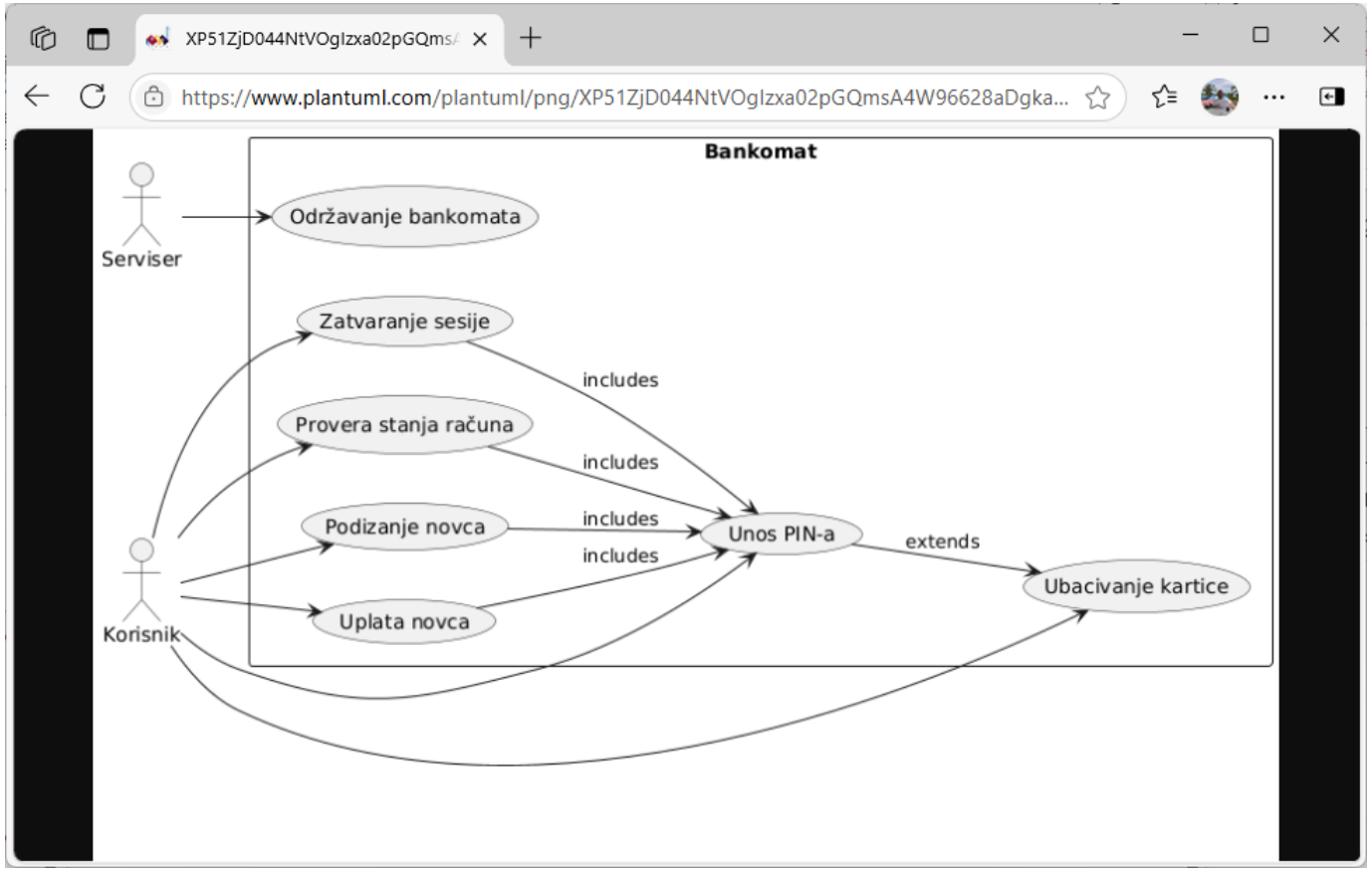


Čuvanje dijagrama



Klikni na dugme PNG

Generisana slika



Desni klik na sliku i opcija Save Image As...

https://chatgpt.com/

- Ukoliko niste zadovoljni generisanim dijagramom ponovo se obratite ChatGPT-u
- Npr: Izbaci aktera serviser ubaci novog aktera Racunar banke koji ažurira stanje na racunu nakog transakcije. Generisi Plant UML kod.
- Generisani kod ponovo ubacite u PlanUML web editor i kliknite na dugme Submit (možete preko prethodnog koda, ali pazite da ne ostane stari kod).

Generisani use case