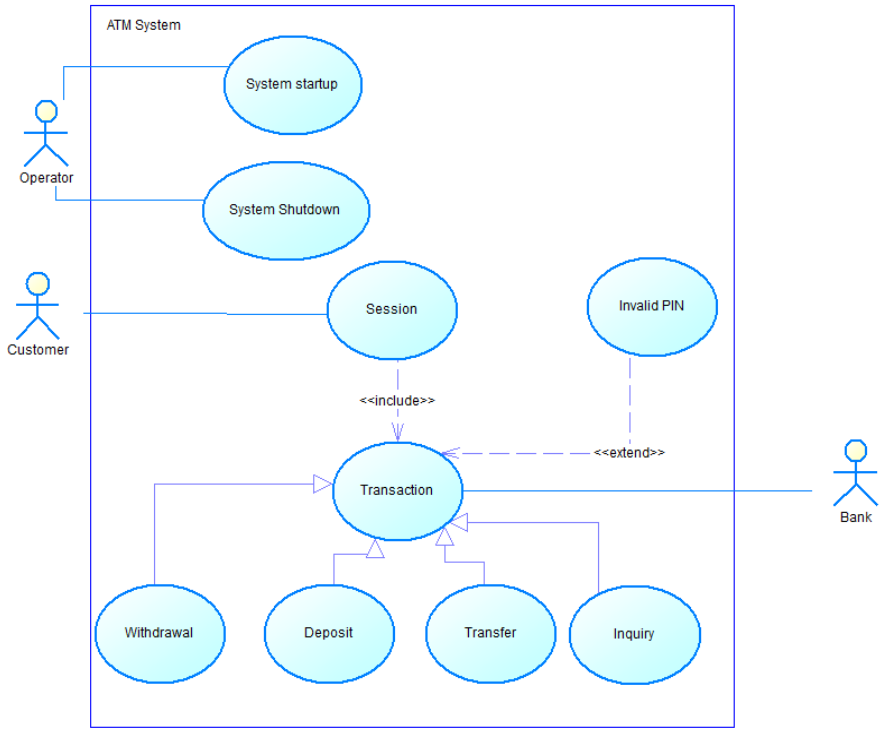
Use Case – dijagram korisničkih funkcija

1. Osnovni koncepti

Nijedan sistem ne postoji izolovano bez interakcije sa ljudskim ili automatizovanim izvođačima, koji taj sistem koriste iz određenih razloga. Izvođači koji koriste određeni sistem očekuju od tog sistema da se ponaša na predvidljive načine.

Korisničke funkcije se primenjuju da se prikaže željeno ponašanje sistema koji se razvija, a da se pri tome ne mora definisati kako se to ponašanje realizuje. Korisničke funkcije obezbeđuju način da se projektanti sporazumevaju sa krajnjim korisnicima sistema i ekspertima iz posmatrane oblasti.

U UML-u se dijagrami korisničkih funkcija koriste za vizualizaciju sistema, tako da korisnici mogu shvatiti kako da koriste sistem, odnosno da projektanti mogu realizovati posmatrani sistem. Dobro struktuirane korisničke funkcije prikazuju samo bitne pojedinosti u ponašanju sistema i nisu ni preterano uopštene ni preterano detaljne.



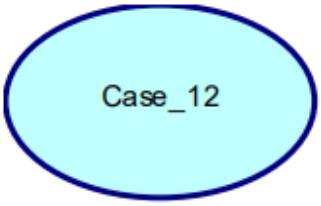
Na primer, da bi se odredilo ponašanje bankomata navodeći u korisničkim funkcijama kakva uzajamna dejstva sa korisnicima treba da postoje, nije potrebno ništa znati o unutrašnjosti sistema. Korisničke funkcije definišu željeno ponašanje, a ne diktiraju kako to ponašanje treba realizovati. Na taj način omogućena je komunikacija između krajnjih korisnika i eksperata iz raznih domena sa projektantima sistema (koji treba da naprave sistem koji ispunjava zahteve bez preteranog ulaženja u detalje).

2. Elementi dijagrama korisničkih funkcija

Dijagrami korisničkih funkcija se obično sastoje od:

- Korisničke funkcije (Use Case)
- Uloge, izvođače (Actor)
- Relacije generalizacije , asocijacije i zavisnosti
- Komentare (Note)
- Pakete (Package)
- Granice sistema

2.1. Korisničke funkcije (Use Case)



Korisnička funkcija predstavlja funkcionalan zahtev posmatranog sistema i izvršava određenu, sagledivu količinu posla. Iz perspektive datog izvođača, korisnička funkcija radi nešto što je izvođaču korisno. Korisnička funkcija opisuje šta sistem radi ali ne definiše kako to radi. Korisnička funkcija se u UML-u predstavlja kao oblačić (elipsa) sa svojim imenom koje je jedinstveno.

Korisnička funkcija može da ima varijatete. Jedna korisnička funkcija može biti specijalizovana verzija neke druge korisničke funkcije (ostvaruje se relacijom generalizacije), može biti sadržana u nekim drugim korisničkim funkcijama (ostvaruje se relacijom obuhvatanja), ili može proširivati ponašanje nekih drugih korisničkih funkcija (ostvaruju se relacijom proširivanja).

Korisnička funkcija predstavlja određenu funkcionalnost sistema iz perspektive korisnika tog sistema. Korisnička funkcija mora imati ime, koje obično predstavlja glagol.

2.2. Uloge, izvođači (Actor)

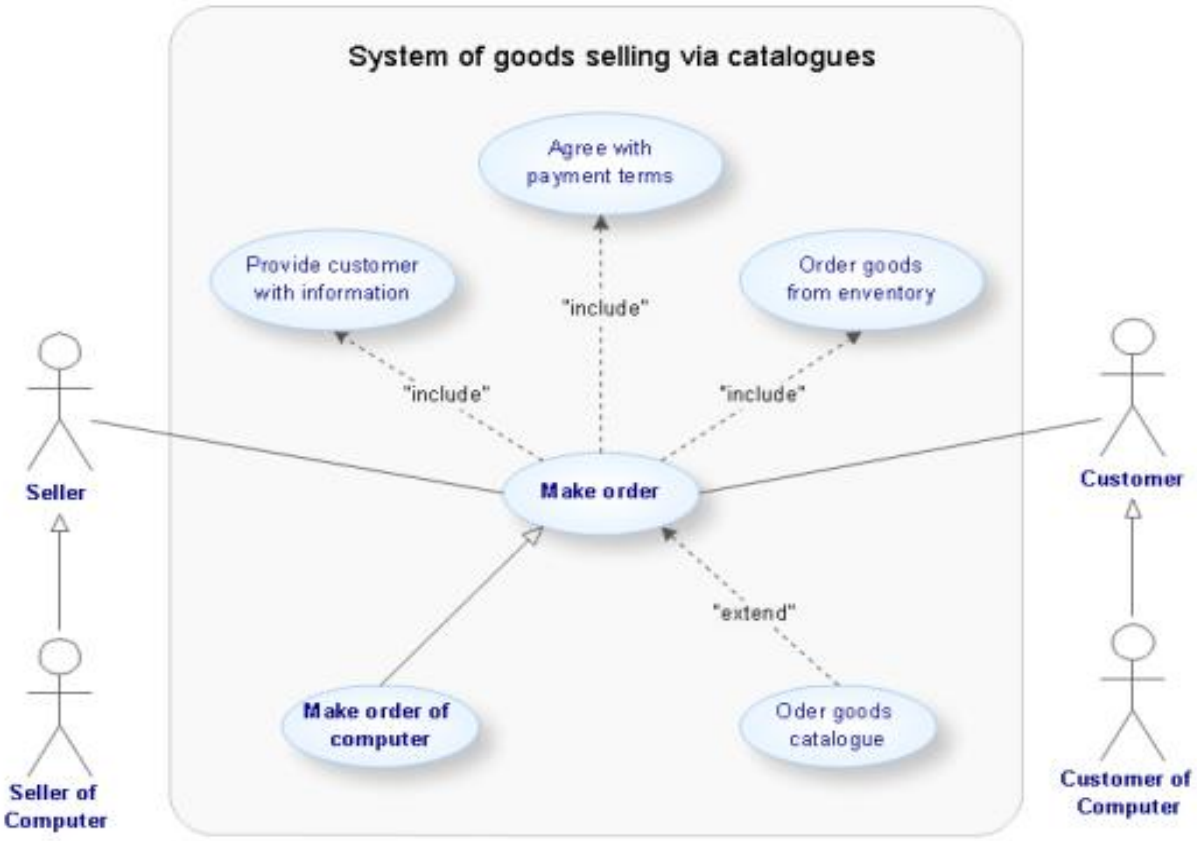
Izvođač predstavlja ulogu koju korisnici korisničkih funkcija izvode kada su u interakciji sa tim korisničkim funkcijama. Uobičajeno je da izvođač predstavlja ulogu koju čovek, hardverski uređaj ili čak neki drugi sistem igra sa posmatranim sistemom. Uloga se u UML-u predstavlja kao ljudska fugurica. Izvođači mogu biti povezani sa korisničkom funkcijom samo pomoću asocijacije.

Da bi se detektovale uloge u posmatranom sistemu:

- Ko/šta će biti interesantan u sistemu?
- Ko/šta želi da utiče na sistem?
- Ko/šta želi da dobija informacije od sistema?
- Ko/šta želi pristup funkcionalnostima sistema?

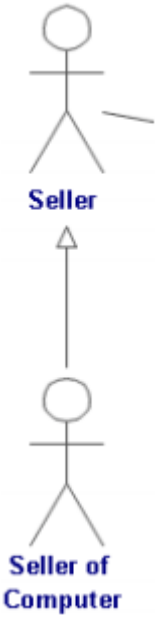
2.3. Relacije generalizacije, asocijacije i zavisnosti

Korisničke funkcije i uloge mogu se organizovati definišući relacije generalizacije, zavisnosti (obuhvatanja – include i proširivanje - extend) i asocijacije između njih.



2.3.1. Relacije generalizacije

Generalizacija (nasleđivanje) između korisničkih funkcija ili između uloga je ista kao generalizacija između klasa. Generalizacija se može koristiti i između uloga i između korisničkih funkcija. Generalizacija mora da ima svoj naziv koji opisuje prirodu veze između uloga ili korisničkih funkcija (PRODAVAC_RAČUNARA_NASLEĐUJE_PRODAVCA, TRANSFER_NOVCA_JE_TRANSAKCIJA...).



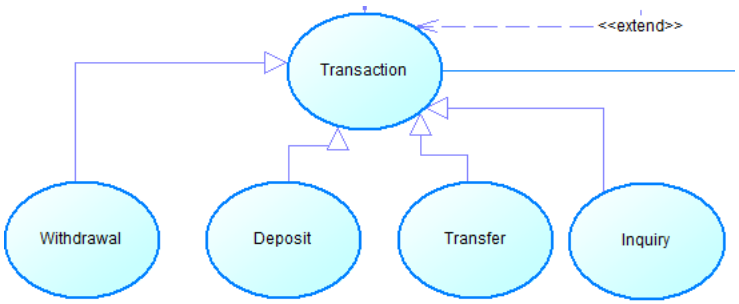
Generalizacija između uloga znači da potomak neke uloge nasleđuje ponašanja i smisao roditeljske uloge; naslednik(dete) može da dopuni ili nadjača ponašanje roditelja; dete može zameniti roditelja na svakom mestu na kome se roditelj pojavljuje.

Sve korisničke funkcije sa kojima je roditelj vezan asocijaciom postaju dostupne i nasledniku (iako nije direktno vezan asocijacijom sa korisničkim funkcijama). (Na datom primeru, sve korisničke funkcije koje može da izvrši prodavac može da izvrši i prodavac računara, iako nije direktno povezan sa tim korisničkim funkcijama).

Generalizacija između korisničkih funkcija znači da potomak neke korisničke funkcije nasleđuje ponašanja i smisao roditeljske korisničke funkcije; naslednik(dete) može da

dopuni ili nadjača ponašanje roditelja; dete može zameniti roditelja na svakom mestu na kome se roditelj pojavljuje.

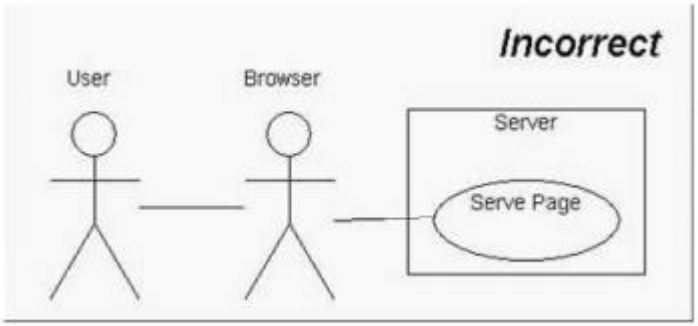
Ukoliko je neka uloga asocijacijom vezana za korisničku funkciju koja ima naslednike, uloga može da izvrši i korisničke funkcije koje nasleđuju datu funkciju, iako uloga i naslednici korisničkih funkcija nisu direktno vezani.



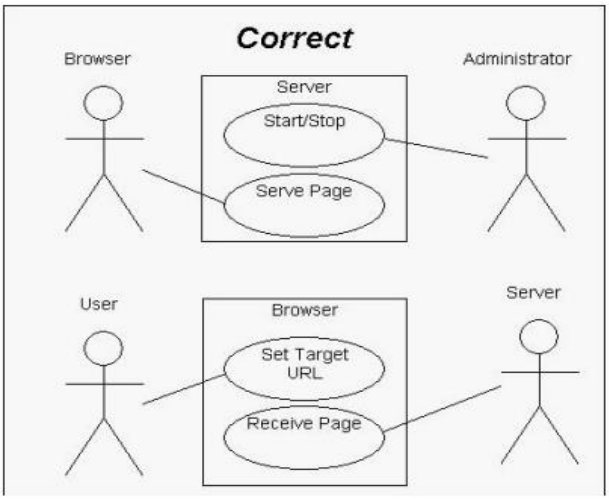
2.3.2. Relacije asocijacije

Relacija asocijacije označava interakciju uloge (izvođača) sa korisničkom funkcijom. Relacija asocijacije predstavlja jedini način povezivanja uloge i korisničke funkcije. Relacija asocijacije treba da ima naziv koji označava funkciju koju izvodi uloga (KUPAC_PRAVI_PORUDŽBU,PRODAVAC_PRIMA_PORUDŽBU, OPERATOR_STARTUJE_SISTEM).

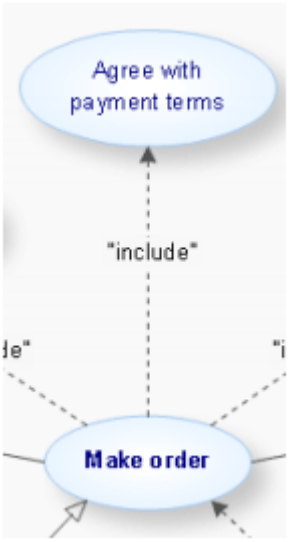
Ukoliko bi postojala interakcija između uloga u posmatranom sistemu, interakciju između uloga ne bismo mogli predstaviti na istom dijagramu (slika levo).



Ono što bi u tom slučaju trebali napraviti jeste kreiranje još jednog Use Case dijagrama, tretirajući jednu ulogu kao poseban sistem, a originalni sistem (sa preostalom ulogom u njemu) kao jednu ulogu u novokreiranom sistemu (slika ispod).



2.3.3. Relacije zavisnosti (include i extend)

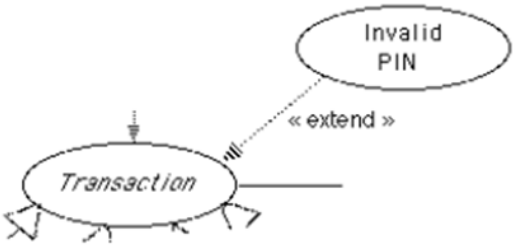


Relacija obuhvatanja – može se primeniti samo između korisničkih funkcija. ciju. Označava se kao zavisnost koristeći stereotip **include**.

Relacija obuhvatanja između korisničkih funkcija znači da osnovna korisnička funkcija eksplicitno obuhvata ponašanje druge korisničke funkcije. Obuhvaćena korisnička funkcija nikad ne može da stoji sama za sebe, već se javlja kao deo neke veće korisničke funkcije koja je obuhvata.

Relaciju obuhvatanja treba koristiti da bi se izbeglo da se isti tok događaja opisuje više puta, stavljajući zajedničko ponašanje u posebnu korisničku funkciju.

Relacija proširivanja - može se primeniti samo između korisničkih funkcija. Označava se kao zavisnost koristeći stereotip **extend**.



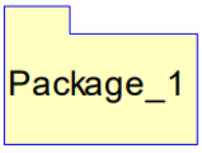
Relacija proširivanja znači da osnovna korisnička funkcija indirektno ugrađuje ponašanje druge korisničke funkcije na mestu koje je indirektno definisano proširujućom korisničkom funkcijom. Osnovna korisnička funkcija može stajati sama, ali pod određenim okolnostima, njeno ponašanje može biti prošireno ponašanjem druge korisničke funkcije. Osnovna korisnička funkcija može biti proširena samo na određenim mestima koja se nazivaju tačke proširenja. Proširivanje se može shvatiti kao da proširujuća korisnička funkcija ubacuje ponašanje u osnovnu korisničku funkciju ukuliko su zadovoljeni određeni uslovi.

Relacija proširivanja koristi se za modelovanje dela korisničke funkcije koji korisnik može videti kao opciono ponašanje. Na taj način razdvaja se opciono od obaveznog ponašanja.

2.3.4. Komentare (Note)

Komentar predstavlja osnovno anotaciono sredstvo koje se može uvrstiti u UML dijagrame. Komentari se obično koriste za dopunjavanje dijagrama sa ograničenjima ili napomenama koji se najbolje izražavaju neformalnim ili formalnim tekstom.

2.3.5. Pakete (Package)



Elementi na dijagramima korisničkih funkcija mogu se organizovati grupisanjem u pakete, na isti način na koji se mogu organizovati klase. Paket je mehanizam opšte namene za organizovanje elemenata u grupe.

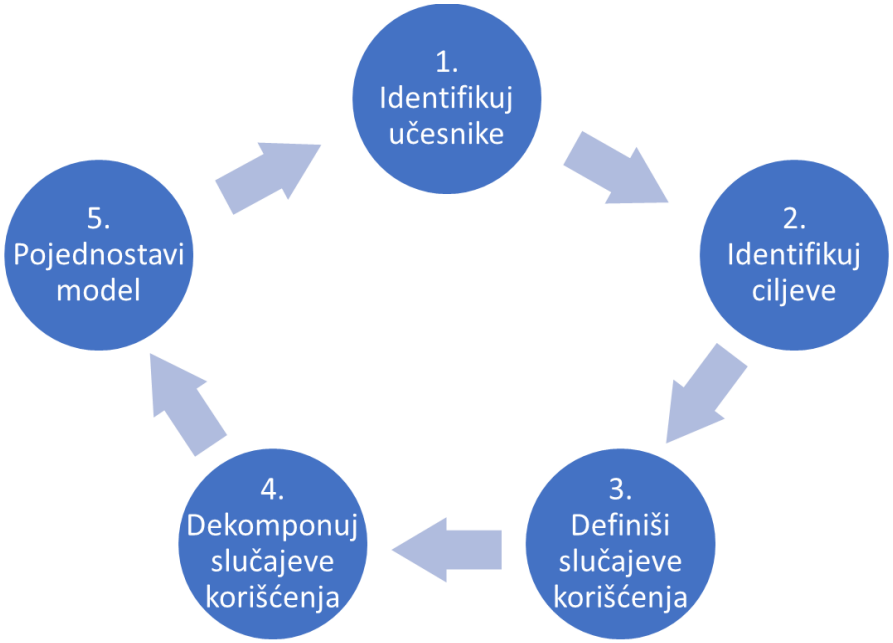
2.3.6. Granice sistema

Pravougaonik oko korisničkih funkcija označava opseg sistema - korisničke funkcije unutar pravougaonika predstavljaju funkcionalnost koju sistem implementira.

3. Use Case modelovanje

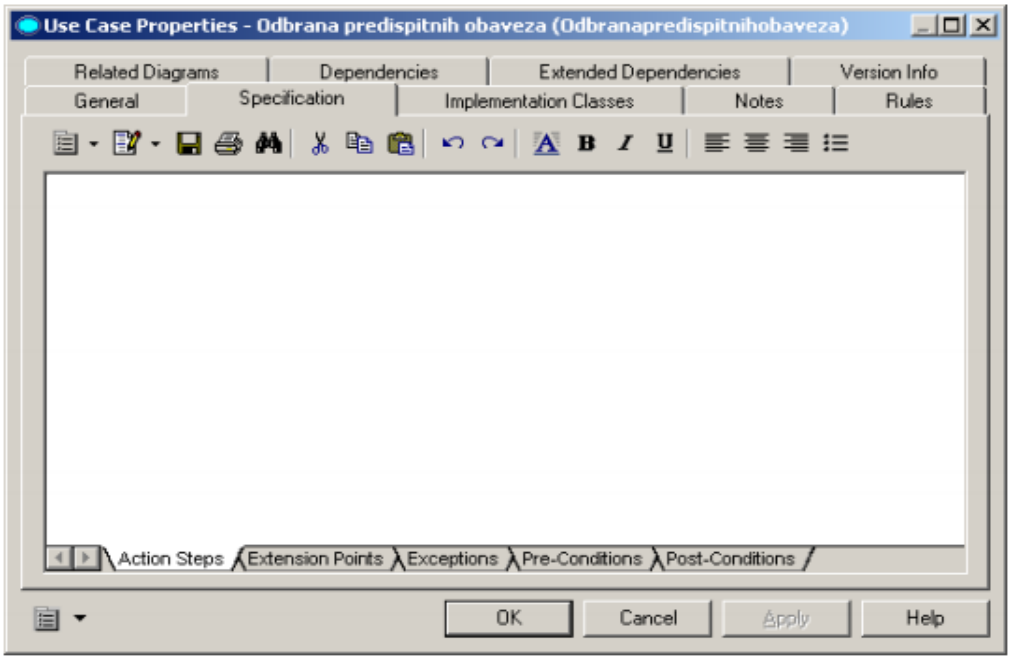
Da bi se modelovalo ponašanje posmatranog sistema:

1. **Identifikujte učesnike (izvođače) -** Koji *akteri* su u interakciji sa posmatranim sistemom? Kandidati za uloge su grupe koje zahtevaju određeno ponašanje radi izvođenja svojih zadataka ili koji su neophodni direktno ili indirektno radi izvršavanja funkcija tog sistema. Organizujte uloge identifikujući opšte i specijalne uloge (primena generalizacije kod uloga).
2. **Identifikuj ciljeve -** Šta sistem treba da radi da ispuni ciljeve učesnika? Da li je potrebno da se vrši čitanje, izmena, kreiranje, uništavanje informacija ili nešto drugo? Koji su događaji koje učesnik treba da dobije ili evidentira?
3. **Definiši slučajeve korišćenja –** Na osnovu ponašanja koje izvođači očekuju ili zahtevaju od sistema, kreirati slučajeve korišćenja. U početku praviti opšte slučajeve, a kasnije ih razraditi na manje. Ako je sistem velik, uvesti pakete za grupisanje elemenata dijagrama.
4. **Dekomponuj slučajeve korišćenja -** Organizujte korisničke funkcije primenjujući relacije obuhvatanja (*include*) i proširivanja (*extend*) radi razlaganja na zajedničke delove ponašanja i razlikovanje izuzetnih ponašanja.
5. **Pojednostavi model –** Ukloniti sve slučajeve korišćenja koji nisu značajni za razumevanje sistema. Isto tako, ukloniti sve izvođače koji nisu u asocijaciji ni sa jednim slučajem korišćenja. Rasporediti preostale elemente tako da se broj linija koje se seku svedu na najmanju moguću meru. Voditi računa o tome da sve korisničke funkcije, uloge, i veze imaju ima koja ukazuju na njihovu namenu.



4. Specifikacija korisničke funkcije

Korisnička funkcija opisuje šta sistem radi, ali ne definiše kako to radi. Moguće je specifikovati ponašanje korisničke funkcije tekstualno opisujući tok događaja, dovoljno jasno da to lako razume i neko ko je neupućen. Kada se piše tok događaja, potrebno je naznačiti kada i kako korisnička funkcija započinje i završava, kada korisnička funkcija stupa u interakciju sa izvođačima, koji se objekti razmenjuju, potrebno je opisati osnovni i alternativne tokove ponašanje.



Specifikaciju korisničke funkcije moguće je odraditi u kartici „Specification“ na korisničkoj funkciji:

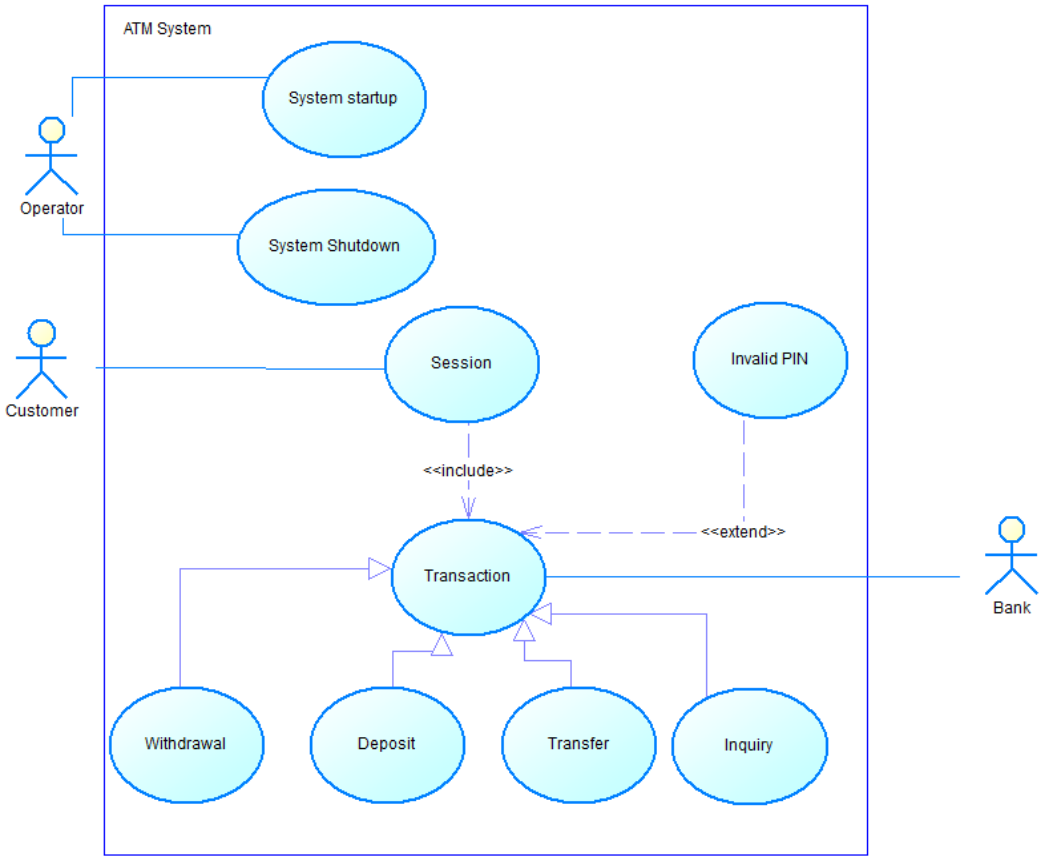
1. **Action steps -** opisuje se *osnovni (uspešni) tok događaja*. Pod osnovnim tokom događaja navodi se kada korisnička funkcija počinje svoje izvršavanje i ko inicira njeno izvršavanje. Ukoliko korisnička funkcija uključuje neke druge funkcije (postoji include zavisnost), navodi se kada se uključena korisnička funkcija izvršava.
2. **Extension Points -** *alternativni tok događaja*, ukoliko korisnička funkcija ima proširenja ona se ovde navode, uz potrebne uslove izvršavanja.
3. **Exceptions -** neuspešni tok događaja.
4. **Pre-Conditions -** navođenje potrebnih *preduslova* da bi se korisnička funkcija uopšte izvršila.

   Post-Conditions - *posledice* uspešnog i/ili neuspešnog izvršavanja korisničke funkcije**.**

Zadatak 1. – koristeći *power designer* alat nacrtati dijagram slučajeva upotrebe za proizvoljan sistem ili deo sistema (npr. bankomat, samoposlužna kasa).

Zadatak 2. – nacrtati *use case* dijagram za deo projektnog zadatka.

Primer 1



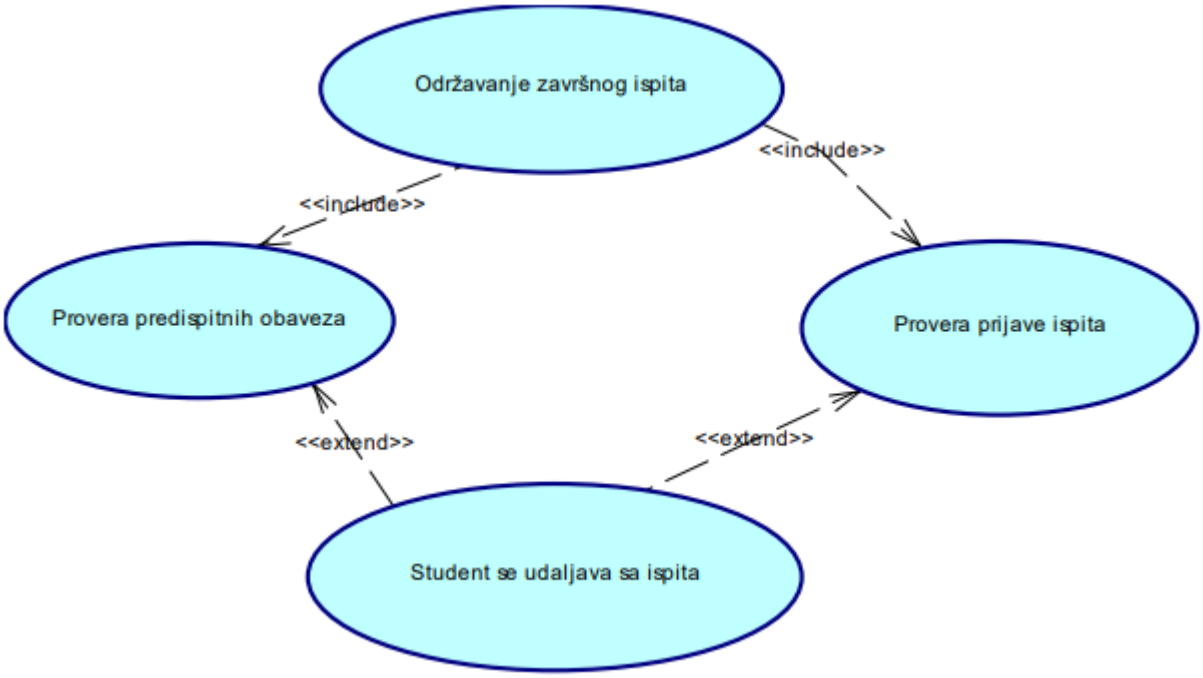
Specifikacija korisničke funkcije „Transaction“

a. Action Steps
    1. Korisnik inicira transakciju ubacujući karticu u bankomat
    2. Korisnik bira jedan od ponuđenih jezika na kome će se izvršiti transakcija
    3. Korisnik unosi svoj PIN code
    4. Sistem vrši proveru validnosti unetog PIN code-a
    5. Korisnik bira jednu od ponuđenih transakcija
b. Extension Points
    1. U slučaju da je korisnik uneo neodgovarajući PIN code prelazi se na slučaj korišćenja „Invalid PIN“
c. Exceptions
    1. Korisnik je tri puta uneo neispravan PIN code, kartica je oduzeta a transakcija otkazana
d. Pre–Conditions
    1. Bankomat je uključen
    2. Korisnik je uneo ispravan PIN code
e. Post – Conditions
    1. Korisnik je uspešno izvršio željenu transakciju

Specifikacija korisničke funkcije „Withdrawal“

a. Action Steps
    1. Korisnik bira jedan od ponuđenih iznosa koji želi da podigne
    2. Korisnik potvrđuje izabrani iznos
    3. Sistem vrši proveru da li postoji dovoljan iznos na računu korisnika
    4. Sistem vrši provera da li u bankomatu postoje dovoljna sredstva
    5. Korisnik bira da li želi štampanje izveštaja o izvršenoj transakciji
b. Extension Points
    1. U slučaju da korisnik nije izabrao nijedan od ponuđenih iznosa, nudi mu se mogućnost da sam unese željeni iznos
    2. U slučaju da je korisnik izabrao opciju da se štampa izveštaj o izvršenoj transakciji, nakon izvršene transakcije štampa se izveštaj
c. Exceptions
    1. Korisnik nema dovoljan iznos sredstava na svom računu
    2. U bankomatu ne postoji dovoljan iznos sredstava za isplatu
d. Pre–Conditions
    1. Bankomat je uključen
    2. Korisnik je uneo ispravan PIN code
e. Post – Conditions
    1. Korisnik je uspešno podigao novac i dobio izveštaj o izvršenoj transakciji

Primer 2



Specifikacija korisničke funkcije „Održavanje završnog ispita“

a. Action Steps
    1. Profesor dolazi na zakazani termin održavanja ispita
    2. Student dolazi na zakazani termin održavanja ispita
    3. Profesor vrši proveru predispitnih obaveza studenta (korisnička funkcija „Provera predispitnih obaveza“)
    4. Profesor vrši proveru prijave ispita studenta (korisnička funkcija „Provera prijave ispita“)
    5. Profesor daje studentu ispitna pitanja
    6. Student po završetku ispita predaje profesoru svoje rešenje ispita

b. Extension Points
    --
c. Exceptions
    1. Student je došao na ispit a nije položio predispitne obaveze
    2. Student je došao na ispit a nije prijavio ispit
    3. Student je prekršio pravila održavanja ispita i udaljen je sa ispita
d. Pre–Conditions
    1. Profesor je došao na zakazani termin održavanja završnog ispita
    2. Barem jedan student je došao na održavanje završnog ispita
e. Post – Conditions
    1. Student je predao rešenje ispita