Oblikovanje programske potpore

Završni ispit

30. siječanj 2018.

Izjavljujem da tijekom izrade ove zadaće neću od drugoga primiti niti drugome pružiti pomoć, te da se neću koristiti nedopuštenim sredstvima. Ove su radnje teška povreda Kodeksa ponašanja te mogu uzrokovati i trajno isključenje s Fakulteta. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

JMBAG	*Ime i prezime*	*Vlastoručni potpis*

Minimum za prolaz završnog ispita je 12 bodova, maksimum 36 bodova **GRUPA A**

1. (1 bod) Navedite značajke dobrog programskog proizvoda.

   Rj. Programski proizvod mora osigurati traženu funkcionalnost i performanse, te mora biti prihvatljiv korisniku, pouzdan i mora se moći održavati.

2. (1 bod) Navedite metode izlučivanja korisničkih zahtjeva.

   Rj. intervjuiranje, scenarij, obrasci uporabe, dinamičke interakcije korištenjem sekvencijskih dijagrama.

3. (1 bod) Navedite tri osnovna Scrum artefakta.

   Rj. Projektni dnevnik zaostataka (Product Backlog), Sprint dnevnik (Sprint backlog) i inkrement.

4. (1 bod) Navedite klasifikaciju arhitekture programske potpore po dosegu.

   Rj. Koncepcijska, logička, izvršna.

5. (1 bod) Objasnite princip dobrog oblikovanja programske potpore: oblikuj konzervativno (engl. *design defensively*).

   Rj. Ne koristiti pretpostavke kako će netko upotrebljavati oblikovanu komponentu obraditi sve slučajeve u kojima se komponenta može neprikladno upotrijebiti provjeriti valjanost ulaza u komponentu provjerom definiranih pretpostavki.

6. (2 boda) Razred `BankovnaUsluga` i sučelje `IRed` opisani su sljedećim kôdom:

```
/**
 * Razred BankovnaUsluga
 */
public abstract class BankovnaUsluga implements IRed {

    private int brojURedu = 0;

    public static final int MAKS_RED = 10;

    public abstract void uciniUslugu (int [] parametri );

    public boolean dodajZadnjeg()
    {
        if (brojURedu == BankovnaUsluga.MAKS_RED) return false;
```

```
            brojURedu = brojURedu + 1;
            Ispis("novi dosao, sad ih je "+brojURedu);  //metoda za ispis na konzolu
            return true;
        }
        public boolean ukloniPrvog()
        {
            if (brojURedu == 0) return false;
            brojURedu = brojURedu - 1;
            Ispis("prvi otisao, ostaje ih "+brojURedu);  //metoda za ispis na konzolu
            return true;
        }
        ...
}


/**
 * Sučelje IRed
 */
public interface IRed {

    public boolean dodajZadnjeg ();

    public boolean ukloniPrvog ();

}
```

a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:

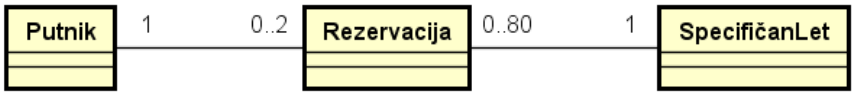
```
IRed red = new BankovnaUsluga();
```

Rj. Nije ispravan, budući da se ne može instancirati primjerak apstraktnog razreda BankovnaUsluga.

b) (1 bod) Implementira li ispravno razred `BankovnaUsluga` sučelje `IRed`? Ukratko objasnite odgovor.

Rj. Da, implementacija sučelja je ispravna, budući da za obje metode definirane u sučelju postoji valjana implementacija.

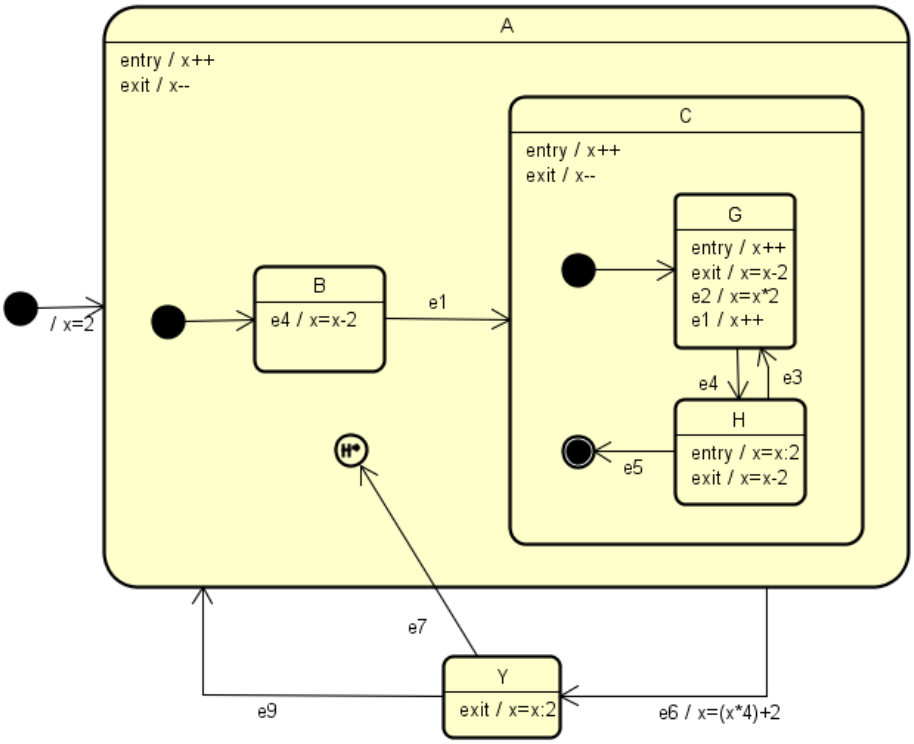
7. (1 bod) Prema donjem dijagramu razreda, odgovorite: koliki je najmanji, a koliki je najveći broj rezervacija prisutnih u sustavu, ako u sustavu postoje 5 putnika i 2 specifična leta?



Rj. Svaki od 5 putnika može imati između 0 i 2 rezervacije. Najmanji broj rezervacija u sustavu je stoga **0**, a najveći **10.**

Obrazloženje (studenti ne moraju navoditi): Budući da je svaka rezervacija povezana s točno jednim specifičnim letom, a svaki specifični let može imati od 0 do 80 rezervacija, to se može dogoditi da sve rezervacije budu na istom letu. Ograničenje je na strani Putnik - Rezervacija, budući da 2 specifična leta mogu imati između 0 i 160 rezervacija. Međutim, ako postoji više od 160 rezervacija u sustavu, broj putnika u sustavu bi morao biti veći, što ovdje nije slučaj.

8. (2 boda) Za zadani UML-dijagram stanja odredite vrijednost varijable x po završetku izvođenja događaja (ulazna vrijednost je **x = 2**), ako je redoslijed događaja koji se izvode: **e1, e2, e2, e6, e7.** Navedite akcije!



Rj. x = (izmjene redom): 3 (entry A), 4 (e1-> entry C), 5 (entry G), 10 (e2), 20 (e2), 18 (exit G zbog e6), 17 (exit C zbog e6), 16 (exit A zbog e6), 66 (e6), 33 (exit Y zbog e7), 34 (entry A), 35 (entry C), 36 (entry G) (konačno).

9. (1 bod) Opišite uvjete kretanja znački (engl. *token*) u UML dijagramu aktivnosti (engl. a*ctivity diagram*).

Rj. dio semantike bez grafičkog prikaza. Značka može predstavljati: upravljački tijek; objekt; podatak. Kreću se od izvorišta prema odredištu vezama ovisno o: ispunjenim uvjetima izvornog čvora, postavljenim uvjetima veza (engl. Edge guard conditions), preduvjetima ciljnog čvora.

10. (1 bod) UML dijagramom razreda prikazan je apstraktni razred klijenta s navedenim stereotipima metoda u radnom okviru klijentsko-poslužiteljske arhitekture. Koje metode programer može ali i ne mora implementirati? Čemu služi ta vrsta metoda?

Rj. connectionEstablished(), connectionClosed(), connectionException().

To su metode koje se mogu po potrebi redefinirati. U radnom okviru za njih postoji minimalna implementacija.

11. (1 bod) Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?

Rješenje: Otkrivanje informacija o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema ispitivane programske podrške.

Priznaje se i kraće rješenje: pronalaženje pogrešaka.

12. (2 boda) Za funkcijsko ispitivanje ekvivalentnim particijama (engl. *equivalence partition*):

a) (1 bod) Navedite korake ispitivanja.

b) (1 bod) Na primjeru jednog ulaznog 4-znamenkastog broja u intervalu [7000, 9997], navedite potrebne minimalne ispitne slučajeve.

1. Odredi particije za sve ulazne varijable.
2. Za sve particije odaberi vrijednosti ispitivanja.
3. Definiraj ispitne slučajeve koristeći odabrane vrijednosti.
4. Odredi očekivane izlaze za odabrane ispitne slučajeve i provedi ispitivanje.

Ispitni slučaj	ulazna vrijednost	očekivani izlaz
IS1	6999	ne prolazi
IS2	7000	prolazi
IS3	8001	prolazi
IS4	9997	prolazi
IS5	9998	ne prolazi

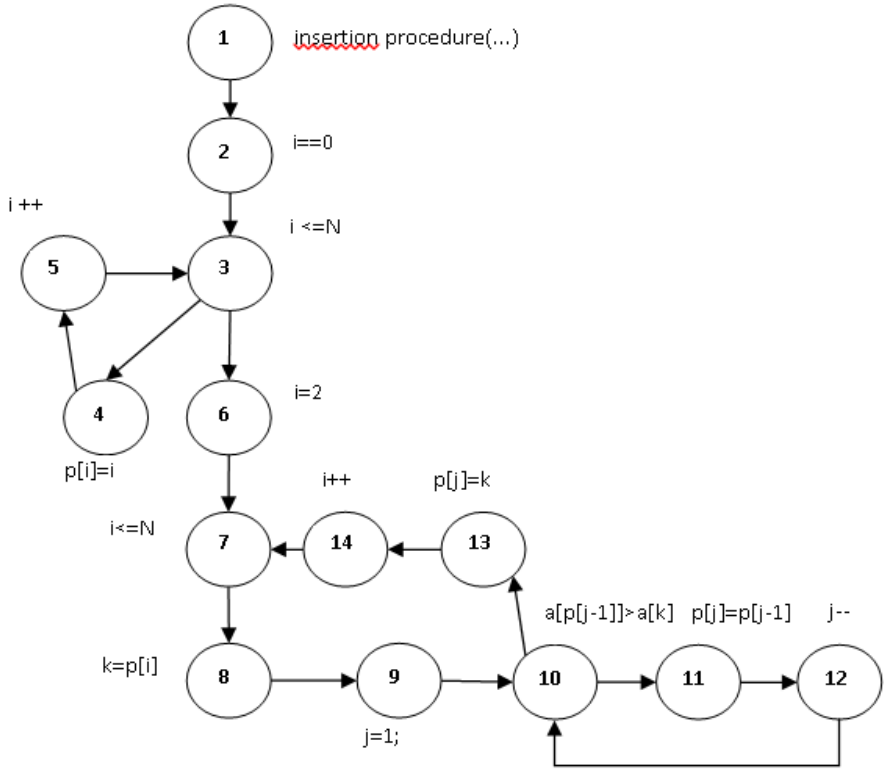
*priznaju se i dodatni ispitni slučajevi: (5000, ne prolazi) (9999, ne prolazi)

13. (2 boda) Za sljedeću funkciju:

a) (1 bod) Nacrtajte graf tijeka programa (engl. *control flow graph*). Uz program navedite odgovarajuće oznake na grafu.

b) (1 bod) Odredite gornju granicu broja ispitnih slučajeva koja jamči potpuno pokrivanje svih naredbi programa. Navedite formulu i izračunajte.

```
01. insertion_procedure (int a[], int p [], int N)
02.  {
03.      int i,j,k;
04.      for (i=0;i<=N; i++)
05.          p[i] = i;
06.      for (i=2;i<=N; i++)
07.        {
08.         k=p[i];j=1;
09.         while (a[p[j-1]] > a[k]) {
10.           p[j] = p[j-1];
11.           j--
12.           }
13.          p[j] = k;
14.        }
15.   }
```



CV(G) = Lukovi – Čvorovi + 2*P =16-14+2=4

14. (1 bod) Navedite osnovno svojstvo razina u općenitoj višerazinskoj arhitekturi (engl. *n-tier architecture*).

Rj. Predstavlja proširenje raspodijeljene arhitekture klijent – poslužitelj. Razina zatvara (skriva, enkapsulira) skup usluga i implementacijske detalje niže razine o kojoj ovisi.

15. (1 bod) Što određuje tijek izvođenja programa u arhitekturi protoka podataka (engl. *data flow*)?

Rj. Redoslijed izvršavanja instrukcija nije određen programskim brojilom već već je nedeterministični, odnosno ovisan o podacima koji su raspoloživi aktorima (engl. functional unit - FUs). Pomakom podataka aktivira se daljnje upravljanje.

16. (1 bod) U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. *version control, revision control, source control*). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?

Rj. u lokalnim repozitorijima kod članova tima, moguć je istodoban rad.

17. (1 bod) Definirajte što znači da je skup logičkih formula Γ konzistentan.

Rj. Skup Γ je *konzistentan* ako ne sadrži formule na temelju kojih bi formule ω_i i $\neg\omega_i$ (istovremeno) bili teoremi (dedukcije).

18. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:

"Na nekom fakultetu postoji samo jedan izborni predmet ‘Ekspertni sustavi’."

Rj.
F(x) = x je fakultet
IP(x,y) = x je izborni predmet na y
=(x,y) = x je jednako y
‘Ekspertni sustavi’ = konst.

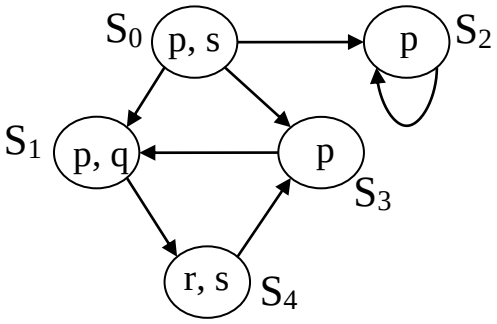
∃x (F(x) ∧ IP(‘Ekspertni sustavi’, x) ∧ ¬∃y (IP(y,x) ∧ ¬=(‘Ekspertni sustavi’,y)))
alternativno:
∃x (F(x) ∧ IP(‘Ekspertni sustavi’,x) ∧ ∀y (IP(y,x) ⇒ =(‘Ekspertni sustavi’,y)))

19. (1 bod) Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic):

"Uvijek nakon req=1 konačno dođe ack=1."

Rj. AG (req=1 ⇒ AF ack=1).

20. (1 bod) Za zadani model implementacije Kripke strukturom M prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu **A** (p **U** r).



Rj. Takva stanja su = S_1, S_3, S_4

Problemski dio **- Parkirni automat**

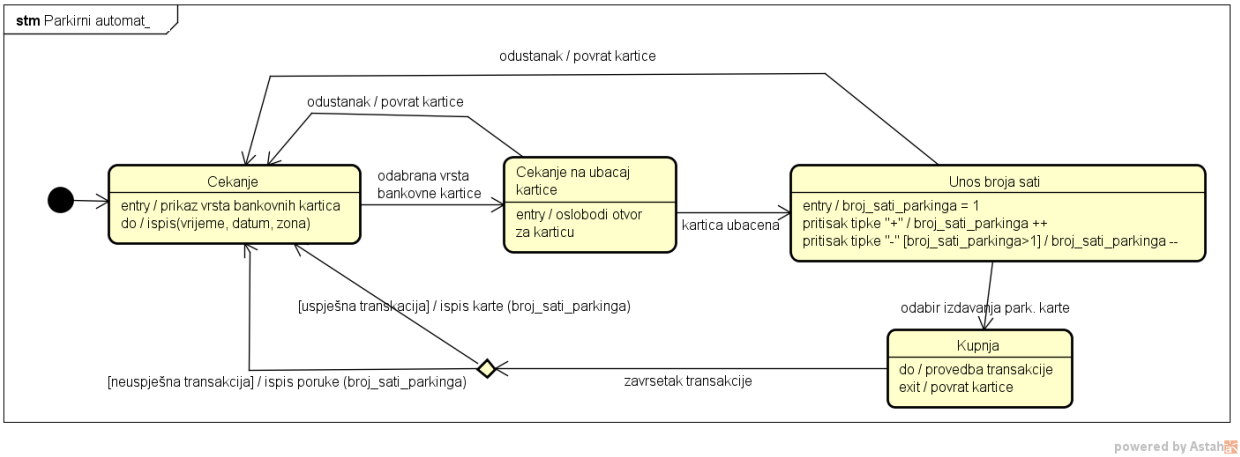
Za prodaju karata na parkiralištima koriste se parkirni automati. Parkirni automat u stanju čekanja na zaslonu prikazuje osnovne informacije i vrste bankovnih kartica kojima je moguće platiti kartu te redovito osvježava vrijeme i datum.
Korisnik započinje kupnju parkirne karte odabirom vrste bankovne kartice kojom želi platiti. Nakon toga parkirni automat oslobađa otvor za ubacivanje kartice i čeka na ubačaj kartice. Kada je kartica ubačena, korisniku se omogućuje unos željenog broja sati parkiranja. Pritiskom na tipku "+" korisnik povećava broj sati parkinga, a pritiskom na tipku "-" smanjuje broj sati parkinga. Na početku broj sati parkinga ima vrijednost "1" te nije moguće smanjiti broj sati na vrijednost manju od "1".
Korisnik u svakom trenutku sve do odabira izdavanja parkirne karte može odustati i parkirni automat u tom slučaju vraća bankovnu karticu.
Odabirom izdavanja parkirne karte korisnik potvrđuje kupnju parkirne karte za odabrani broj sati te se provodi transakcija[1]. Ako je transakcija bila uspješna ispisuje se parkirna karta. Ako transakcija iz bilo kojeg razloga nije uspjela na zaslonu se ispisuje poruka o neuspjeloj transakciji. Istovremeno s ispisom karte ili poruke korisniku se vraća bankovna kartica.

U Centrali sustava naplate parkinga nalazi se poslužiteljsko računalo na kojemu su pokrenuti *poslužitelj poruka*, *poslužitelj baze podataka* i *web poslužitelj*.
Parkirni automati periodički dojavljuju status uporabom XMPP/TLS protokola poslužitelju poruka Centrale, pri čemu koriste ugrađene GSM module za komunikaciju.
Aplikacija za obradu poruka na *poslužitelju poruka* komunicira s bazom podataka Apache Derby, koja se nalazi na poslužitelju baze podataka. Na web poslužitelju nalazi se web aplikacija koja također pristupa bazi podataka. Web aplikacija mora omogućiti administratoru pristup s udaljenog računala putem web preglednika korištenjem sigurne HTTPS veze.

21. (4 boda) Dijagram stanja

UML-dijagramom stanja (engl. *statechart diagram*) modelirajte parkirni automat prilikom uporabe automata od strane korisnika.

Rješenje:



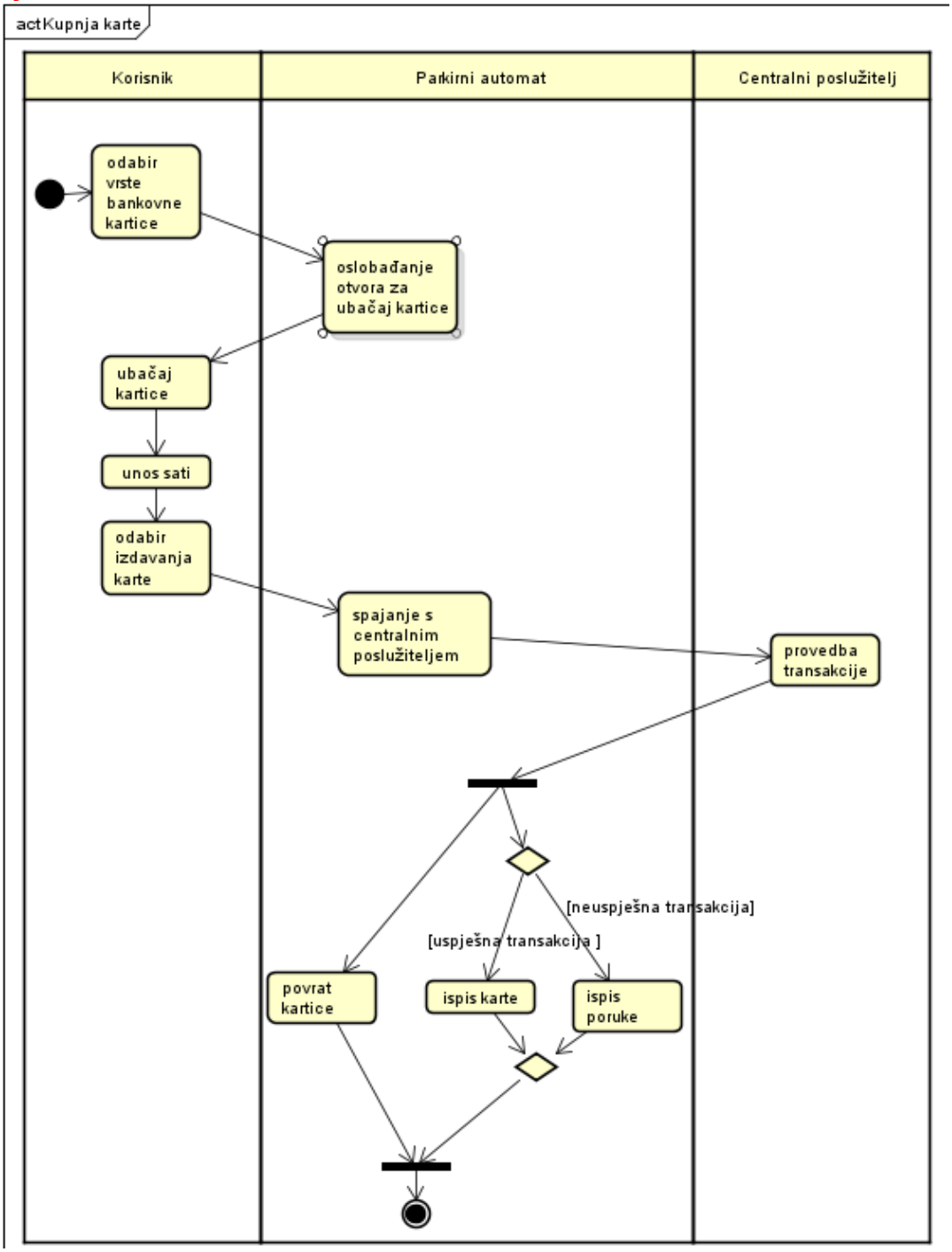
Kriteriji bodovanja:
Prikazana sva stanja automata: 1,5 bod
Prijelazi vođeni događajima: 1 bod
Semantika uvjeta i grananja: 0,5 bod
Ispravna notacija stanja (naziv, varijable stanja i akcije *entry*/*do*/*exit*): 1 bod

[1] NAPOMENA: Transakcija se provodi bez potvrde pinom.

22. (4 boda) Dijagram aktivnosti

UML dijagramom aktivnosti (engl. *activity diagram*) modelirajte kupnju parkirne karte počevši od odabira vrste bankovne kartice. Uzmite dodatno u obzir da se pri provođenju transakcije automat spaja s Centralnim poslužiteljem preko kojeg se obavlja naplata s bankovne kartice.
Nije potrebno modelirati mogućnost odustajanja korisnika od kupnje.

Rj.

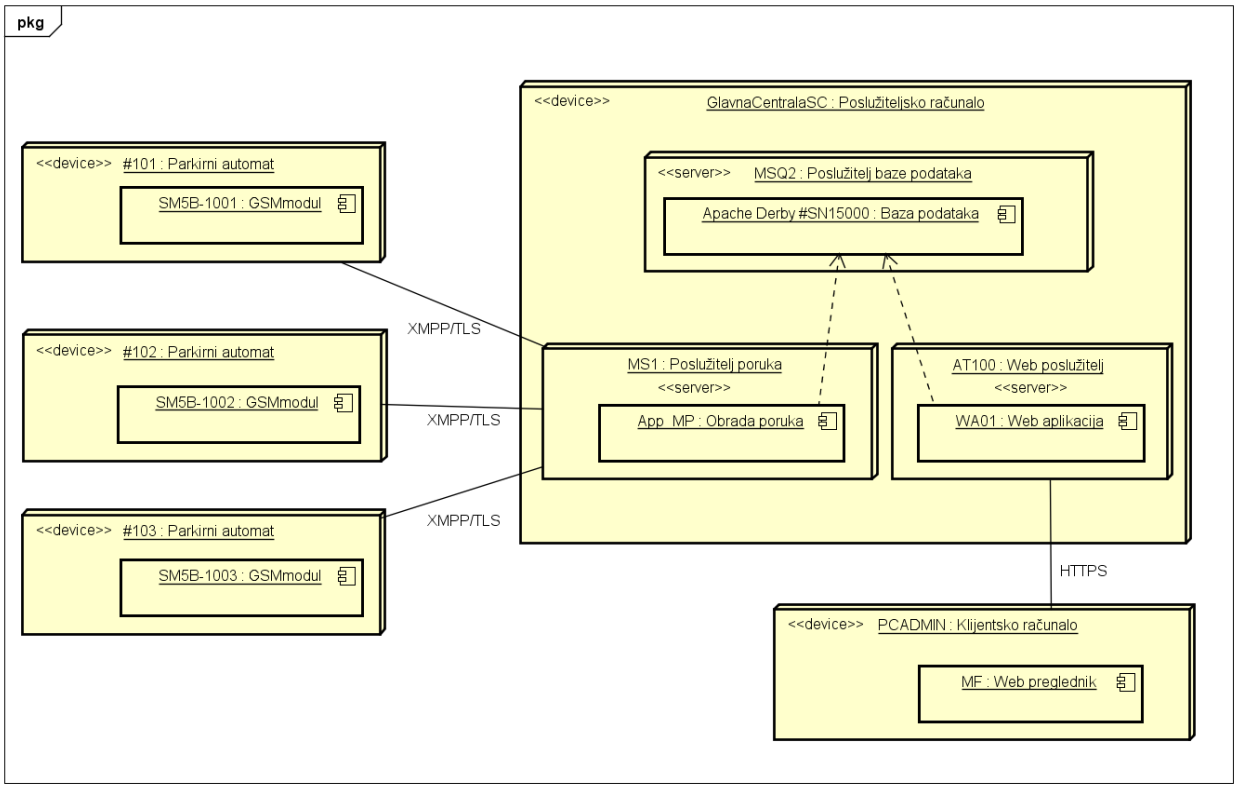


Kriteriji bodovanja:
Prikazane sve akcije: 2,5 boda
Akcije točno pridružene participijama: 1 bod
Semantika uvjeta i grananja: 0,5 bod

23. (4 boda) Dijagram razmještaja

Odgovarajućim UML dijagramom razmještaja instanci (engl. *Instance Level Deployment Diagram*) modelirajte sustav parkirnog automata, pri čemu treba prikazati komunikaciju tri parkirna automata s Centralom. Nazive instanci odredite razumljivo te navedite odgovarajuće stereotipe.

Rj.



Kriteriji bodovanja:
Prikazani svi čvorovi: 1.5 bod
Prikazane sve komponente: 1 bod
Prikazane sve veze između čvorova: 0,5 boda
Prikazane sve veze između komponenti: 0.5 boda
Sve instance ispravno imenovane: 0.5 boda (priznaju se manja odstupanja u nazivima)
Tolerira se ne navođenje stereotipova, ako je jasno što je čvor, a što komponenta.