Oblikovanje programske potpore

Završni ispit

2. veljače 2015.

JMBAG	Ime i prezime	Vlastoručni potpis

Minimum za prolaz je 12 bodova, maksimum 36 bodova **GRUPA B**

1. (1 bod) Koje su osnovne grupe troškova koje ubrajamo u ukupnu cijenu programske potpore?

Rješenje: Cijena programske potpore sastoji se iz troškova razvoja, oblikovanja, ispitivanja i održavanja.

2. (1 bod) Navedite generičke aktivnosti prisutne u različitim modelima inženjerstva zahtjeva.

Rješenje: studija izvedivosti (engl. feasibility study), izlučivanje zahtjeva (engl. requirements elicitation) i analiza i specifikacija zahtjeva, validacija zahtjeva, upravljanje promjenama u zahtjevima

3. (1 bod) Nabrojite barem tri metode koje se koriste pri izlučivanju zahtjeva u oblikovanju programske potpore.

Rješenje: intervju, scenarij, izlučivanje i specificiranje zahtjeva UML obrascima uporabe, specifikacija dinamičkih interakcija -UML sekvencijski dijagrami ...

4. (1 bod) Opišite svojstva konzistentnih zahtjeva u procesu inženjerstva zahtjeva.

Rješenje: Konzistentni zahtjevi ne smiju sadržavati konflikte ili kontradikcije u opisima zahtijevanih mogućnosti.

5. (1 bod) Odgovore na koja pitanja treba definirati cjeloviti proces razvoja programske potpore?

Rješenje: Proces definira TKO radi ŠTO, KADA i KAKO postići željeni cilj.

6. (1 bod) Navedite barem četiri prednosti definiranja arhitekture programske potpore.

Rješenje: Smanjuje cijenu oblikovanja, razvoja i održavanja programskog proizvoda.
Omogućuje ponovnu uporabu rješenja (engl. re-use).
Poboljšava razumljivost.

Poboljšava kvalitetu proizvoda.
Razjašnjava zahtjeve.
Omogućuje donošenje temeljnih inženjerskih odluka.
Omogućuje ranu analizu i uočavanje pogrešaka u oblikovanju.

7. (1 bod) Koji je osnovni cilj primjene principa oblikovanja: "Povećaj uporabu postojećeg"?

Rješenje: Smanjuje trošak i povećava stabilnost (pouzdanost) sustava

može i: što veća aktivna ponovna uporaba komponenti korištenje prethodnih investicija

8. (1 bod) Navedite faze životnog ciklusa unificiranog procesa (engl. *Unified Process*).

Rješenje: Početak (engl. Inception)
Razrada (engl. Elaboration)
Izgradnja (engl. Construction)
Prijenos (engl. Transition

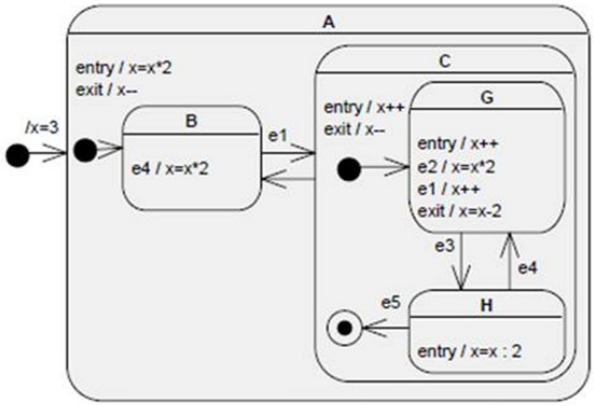
9. (1 bod) Kako nazivamo grupu metoda/ metodu razvoja programske potpore kojima je zajednički iterativni razvoj uz male inkremente i naglaskom na ljude i međusobne odnose?

Rješenje: Ubrzani (agilni) proces (engl. Agile processes), priznaje se i XP, Scrum...

10. (1 bod) Što je metoda u objektno orijentiranim sustavima i kako se prikazuje poziv metode na UML-sekvencijskom dijagramu?

Rješenje: Metoda je način izvođenja ili implementacija neke operacije. To je procedura, funkcija, rutina, proceduralna apstrakcija koja se koristi za implementaciju ponašanja razreda.
Poziv metode prikazuje se strelicom s nazivom poruke (metoda je poziv ili slanje poruka).

11.(1 bod) Za zadani dijagram napišite redoslijed događaja koji se izvode do dolaska u završno stanje uz pretpostavku da je trenutno stanje B.



Rješenje: e1, entry (u C), entry (u G), exit (u G), e3, entry (u H), e5.

Moguće su i varijante koje uključuju događaje e4, e2 i e1 jer zadatak ne spominje najkraći mogući redoslijed događaja.

12.(1 bod) Navedite UML-dijagrame interakcije koji su crtani i komentirani u dokumentaciji projekta iz kolegija Oblikovanje programske potpore.

Rješenje: Sekvencijski i komunikacijski dijagrami.

13.(1 bod) Navedite osnovne korake pri oblikovanju ispitnih slučajeva.

Rješenje: Odabir što ispitivati, Odluka kako ispitivati (npr. black box, white box), Razvoj ispitnih slučajeva, Predikcija rezultata.

14.(1 bod) Tijekom procesa strukturnog ispitivanja (engl. *white box*) modula programske potpore, generiran je graf tijeka programa (engl. *control flow graph*) koji sadrži 10 čvorova i 11 lukova bez dodatnih povezanih komponenti. Odredite najmanji broj potrebnih ispitnih slučaja za sve temeljne putove programa.

Rješenje: $CV(G) = \text{lukovi} - \text{čvorovi} + 2 = 11 - 10 + 2 = 3$

15.(1 bod) Primjenom principa ekvivalencije particija, oblikujte ispitne slučajeve za ispitivanje jedne cjelobrojne varijable s ograničenjem vrijednosti [170 – 9999>. U rješenju odaberite pogodne konkretne vrijednosti ispitivanja i sve minimalne elemente ispitnog slučaja.

Rješenje:

Test	Podatak	Očekivani rezultat
1	100	NE
2	169	NE
3	170	DA
4	1000	DA
5	9998	DA
6	9999	NE
7	20000	NE

16. (2 boda) Navedite definiciju logičke ekvivalencije propozicijskih formula, te definiciju logičke posljedice.

Rješenje:

- Dvije formule su ekvivalentne ako imaju jednaku istinosnu vrijednost za svaku interpretaciju, ili: Formule a i b su ekvivalentne ako je $[(a \Rightarrow b) \wedge (b \Rightarrow a)]$ uvijek istinita (a je logička posljedica od b i b je logička posljedica od a).
- Po definiciji, formula b je logička posljedica formule a ako je svaki model formule a ujedno i model formule b (formula b može imati i druge modele).

17. (2 boda) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:

"Ako na put krećeš radnim danom prije 6:30, idi Maksimirskom, a ako krećeš radnim danom nakon 7:00, idi Branimirovom."

Rješenje: Radni_dan(x) - x je radni dan
T<(x) - kreće u vrijeme prije x
T>(x) - kreće u vrijeme nakon x
Idi_ulicom(x) - ide ulicom x
Konstante: Maksimirska, Branimirova, 6:30, 7:30

$\forall x(\text{Radni\_dan}(x) \wedge T<(6:30) \Rightarrow \text{Idi\_ulicom}(\text{Maksimirska})) \wedge \forall x(\text{Radni\_dan}(x) \wedge T>(7:30) \Rightarrow \text{Idi\_ulicom}(\text{Branimirova}))$

Priznavane su i slične, ali valjane varijante u FOPL-u.

18. (2 boda) Preslikajte izjavu iz prirodnog jezika u CTL formulu koja će ispitati ispravnost tvrdnji te definirajte potrebne propozicijske simbole:

a) "Uvijek vrijedi da dretvu nije moguće terminirati u koraku koji slijedi nakon kreiranja dretve."

Rješenje:
t- dretva je terminirana
c- dretva je kreirana

$$AG(c \Rightarrow AX \neg t)$$

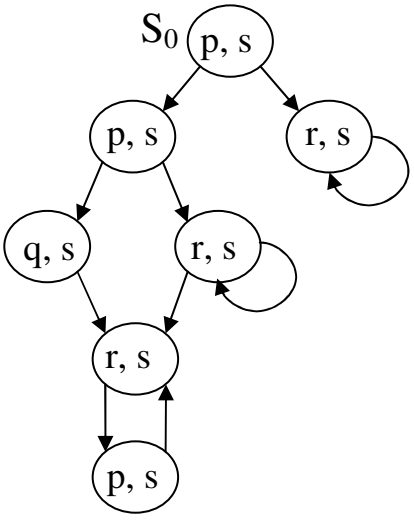
b) "Dretva nikada ne može doći u stanje od kojeg dalje zauvijek vrijedi da je u stanju čekanja".

cekaj – dretva je u stanju čekanja

$$\neg EF\ AG(cekaj)$$

19. (3 boda) Za zadani model implementacije M Kripke strukturom prema slici potrebno je:

a) Dopuniti model i odrediti S (skup stanja), R (relaciju prijelaza), L (funkciju označavanja).
b) Provjerite istinitost tvrdnje M, $\mathbf{S}_0$ |=AG EF **r** uz obrazloženje.
c) Odrediti sva stanja koja zadovoljavaju formulu E (**p** U E(**r** U **q**).



Rješenje:
a) Dopunjuje se sa stanjima: S1,...,S6.
S = {S0, S1, S2, S3, S4, S5, S6}
R = {(S0,S1), (S0,S2), (S2,S2), (S1,S3), (S1,S4), (S3,S5), (S4,S4), (S4,S5), (S5,S6), (S6,S5)}.
L(S0) = {p,s}; L(S1) = {p,s}; L(S2) = {r,s}; L(S3) = {q,s}; L(S4) = {r,s}; L(S5) = {r,s}; L(S6) = {p,s}.

b) EF r zadovoljavaju sva stanja (ili je u njima istinit r ili se može doći do stanja u kojima je istinit r). Prema tome iz stanja S0 vrijedi formula AG EF r.
c) E(r U q) zadovoljava samo stanje u kojem je Q istinit (S3). Do tog stanje moguće je doći iz S0 i S1 u kojima vrijedi p pa je formula E (p U E(r U q)) istinita za stanja S0, S1 i S3.

20. (1 bod) Opišite ulaze i značenje rezultata formalne verifikacija programske potpore metodom provjere modela.

- I = Implementacija, ***S*** =Specifikacija
- ***DA*** = model sustava (implementacija) logički zadovoljava specifikaciju
- ***NE*** - ispis pogrešnog izvođenja programa

Problemski dio- UML Dijagrami

Opći opis: Parkirni automat je namijenjen prodaji parkirnih karata na parkiralištu. Korisnik se služi parkirnim automatom tako što u njega jedan ili više puta ubacuje gotovinu, koja po tipu može biti novčanica ili kovanica. Na prednjoj strani uređaja nalazi se sučelje automata koje se sastoji od:

- o LCD zaslona na kojem je iskazano točno vrijeme, datum, parking zona u kojoj se nalazi uređaj i poruke prema korisniku,
- o dva otvora za ubacivanje novčanica i kovanica,
- o otvora za uzimanje ostatka novca,
- o tipke za unos trajanja parkiranja,
- o tipke za izdavanje karte,
- o tipke za odustajanje,
- o pisača za ispis karata.

Automat komunicira GSM modulom s bazom podataka Centrale, kojoj svakih sat vremena dojavljuje promet. Ako automat otkrije bilo kakvu pogrešku u svojem radu, onda to on također dojavljuje bazi podataka Centrale. Automat se napaja preko akumulatora koji mu daje autonomiju rada od 20 do 30 dana bez dopunjavanja preko solarnih ćelija ili iz električne mreže.

NAPOMENA: Prilikom izrade pojedinih dijagrama uz opći opis parkirnog automata i procesa uzmite u obzir i eksplicitno navedene zahtjeve uz pojedine zadatke.

21. (3 boda) Dijagram obrazaca uporabe

Odgovarajućim dijagramom obrazaca uporabe (engl. *use case diagram*) modelirajte procese plaćanja parkinga i dojave bazi podataka Centrale.

- o Korisnik, nakon ubacivanja gotovine, može unijeti broj sati za koji će platiti parking kao i pokrenuti izdavanje parkirne karte.
- o Parkirni automat obavlja naplatu parkinga po pokretanju izdavanja parkirne karte. Pri obavljanju naplate, u slučaju da korisnik nije unio željeno trajanje parkinga parkirni automat vraća korisniku ostatak novca u odnosu na zaokruženi satni iznos i to u obliku kovanica.
- o U slučaju da naplata nije moguća, jer nije uneseno dovoljno sredstava da pokriju satni iznos, korisniku automat ispisuje poruku „Neuspjela naplata". U svakom drugom slučaju ispisuje se parkirna karta.

22.(4 boda) Dijagram stanja

Odgovarajućim UML-dijagramom stanja (engl. *statechart*) modelirajte unutarnja stanja parkirnog automata.

- o Automat je nakon uključivanja u „Stanju čekanja" (na LCD-u ispisuje vrijeme, datum i zonu) sve dok korisnik ne započne s ubacivanjem novca čime prelazi u stanje „Prihvat novca".

- U stanju „Prihvat novca" ostaje sve dok korisnik ne napravi: (1) odabir sati ili (2) izdavanje parkirne karte. Ukoliko korisnik napravi odabir sati, automat prelazi u novo stanje „Unos sati" u kojem ostaje sve dok se ne odabere izdavanje parkirne karte.
- Odabirom izdavanja parkirne karte izračunava se cijena i ako ima dovoljno sredstava naplaćuje se karta. U suprotnom se na LCD-u ispisuje poruka „Neuspjela naplata" i ponovno omogućuje korisniku ubačaj dodatnog novca.
- Prilikom ubacivanja novca ili odabira sati, korisnik u svakom trenutku može odustati i parkirni automat vraća odgovarajući iznos novca te se vraća u „Stanje čekanja".
- Pretpostavite da u bilo kojem trenutku može doći do pogreške (događaj „detektiran problem") pri čemu automat prelazi u stanje „Kvar" i dojavljuje poruku o pogrešci putem GSM-a bazi podataka Centrale. Pritom se u stanju „Kvara" cijelo vrijeme ispisuje poruka „Automat nije u funkciji". Nakon izvršenog popravka automat se vraća u „Stanje čekanja".

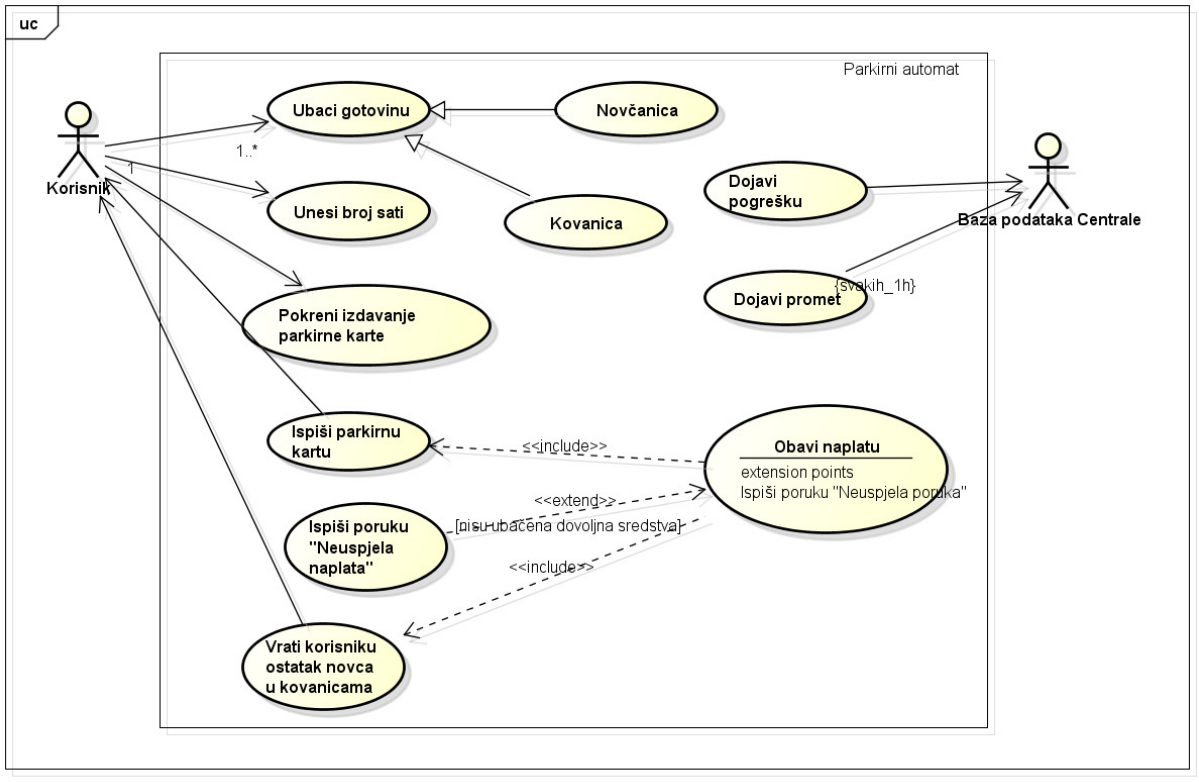
23.(4 boda) Dijagram komponenata

Odgovarajućim UML-dijagramom komponenata (engl. *component diagram*) modelirajte komponente parkirnog automata i sučelja između njih.

- Glavna komponenta parkirnog automata je izvršna datoteka „PParkirni".
- Napajanje parkirnog automata ima programske komponente „Akumulator", „SolarneĆelije" i „ElektričnaMreža" koje posjeduju sučelje „PokreniNapajanje". Akumulator ima dodatna sučelja „DohvatiAutonomiju" i „PokreniNadopunjavanje". Inteligentni algoritmi autonomnog upravljanja podsustavom napajanja parkirnog automata implementirani su u izvršnoj datoteci „PNapajanje" koje je putem sučelja „UpravljajNapajanjem" povezano s izvršnom datotekom „PParkirni".
- Naplata parkirnog automata ima izvršnu datoteku „PNaplata" sa sučeljem „UpravljajNaplatom", te komponentu„ČitačNovčanica" sa sučeljima „DohvatiPotvrduNaplate" i „DohvatiIznosZaNaplatu". Cjenik parkinga u zonama zapisan je u XML datoteci „CjenikParkinga" o kojoj ovisi te se upotrebljava u algoritmima komponente „PNaplata". Izvršna datoteka „PParkirni" komunicira s izvršnom komponentom „PNaplata" koja samostalno obavlja sve ostale funkcije koje nije potrebno modelirati npr. procese poput brojanja gotovine, povrat kovanica i sl.
- Sklopovlje koje koriste sve gore navedene komponente nije potrebno modelirati.
- Označite sučelja komponenti gdje je to potrebno. Ispravno definirajte smjerove veza između komponenti u dijagramu. Grupirajte odgovarajuće komponente u zasebne pakete.

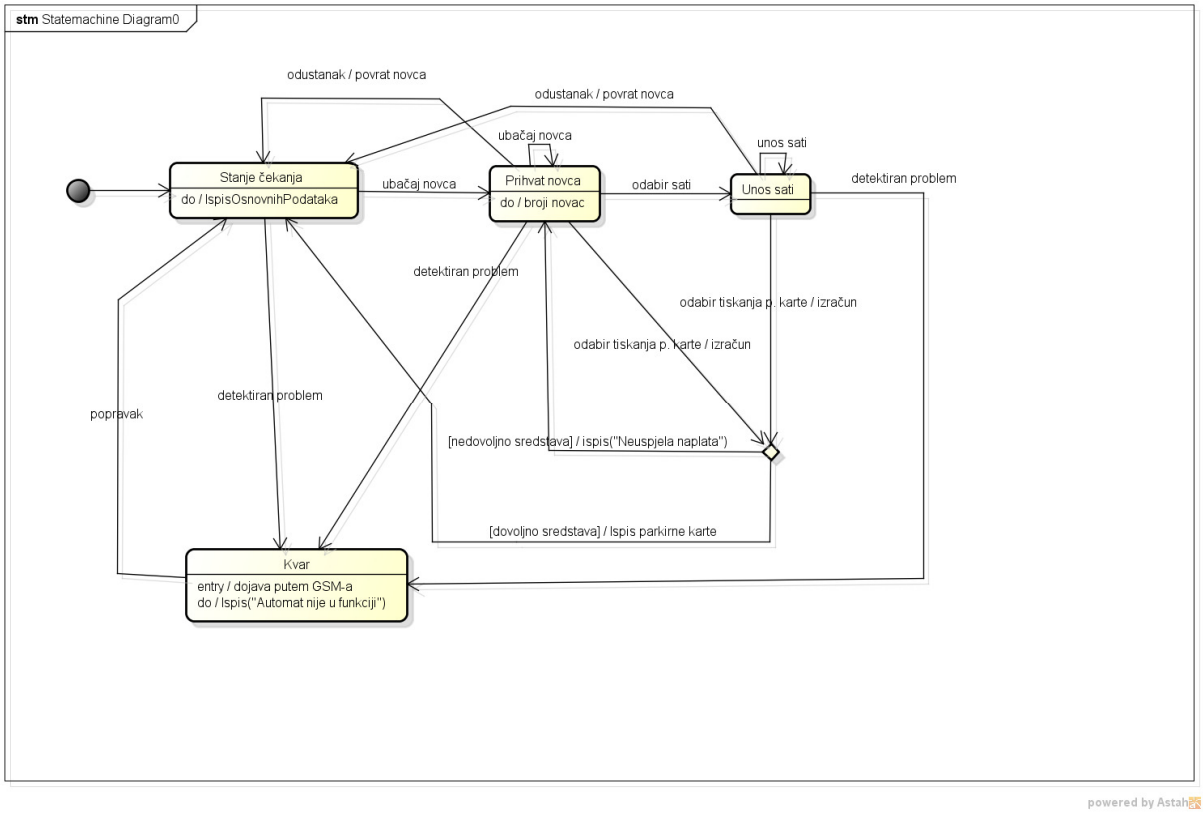
Rješenja:

21.



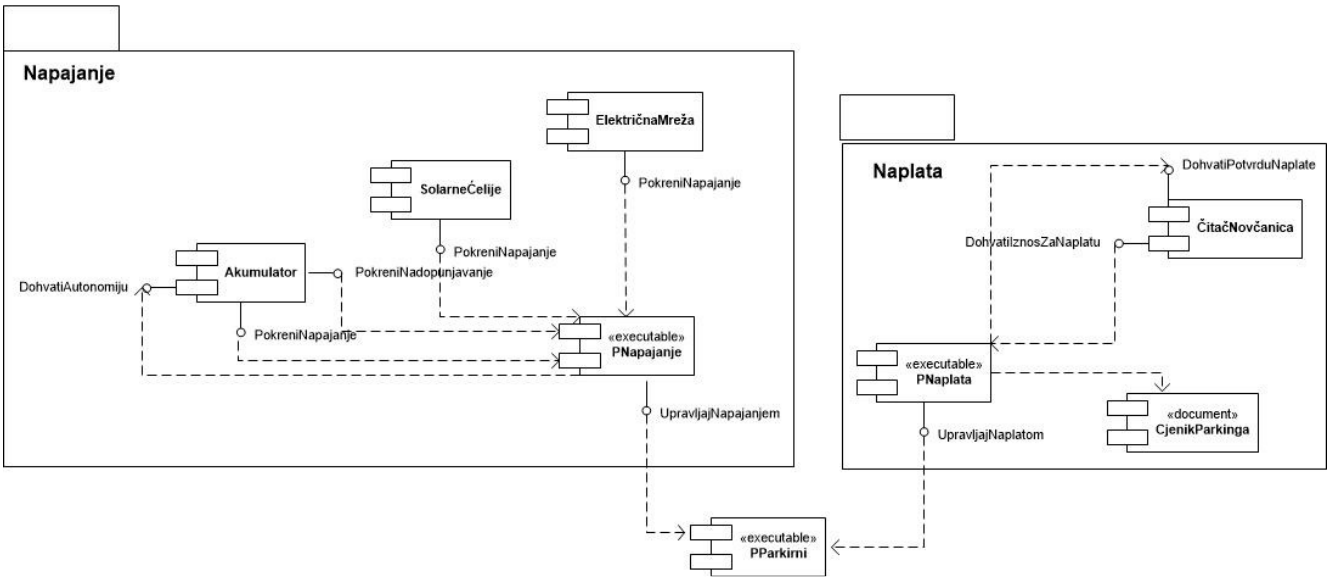
Napomene: - Parkirni automat nije aktor, već modelirani sustav u cjelini koji interagira s aktorima
- Bazu podataka je nužno eksplicitno modelirati jer ne interagira sa svim obrascima uporabe
- Priznaje se i rješenje u kojem je naveden <<extend>> s odgovarajućim uvjetom između "Obavi naplatu" i "Vrati korisniku ostatak novca u kovanicama".
- Priznaje se i rješenje u kojem je naveden <<extend>> s odgovarajućim uvjetom između "Obavi naplatu" i "Ispiši parkirnu kartu".

22.



Napomene: - Parkirni automat nikad ne staje (ne postoji završno pseudo stanje)
- „odabir sati", „Ubačaj novca", „popravak" itd. su događaji koji uzrokuju prijelaze između stanja, a ne uvjeti

23.



Napomene:
Potrebno je razlikovati importirana i eksportirana sučelja (smjer veza).
Potrebno je navesti nazive sučelja.
Potrebno je označavati veze između komponenata i unutar paketa Napajanja i Naplata
Trebaju se koristiti stereotipovi za oznaku tipa komponenti gdje je to potrebno
UML ima jasan simbol za komponentu koji se treba koristiti.
Veze nemaju stereotipove.