Oblikovanje programske potpore

Završni ispit

28. siječnja 2020.

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 B**

1. (1 bod) Navedite generičke aktivnosti programskog inženjerstva.

   Rj. specifikacije, oblikovanje i implementacija, validacija i verifikacija, evolucija.

2. (1 bod) U inkrementalnom pristupu razvoja programske potpore sustav se korisniku ne isporučuje u cjelini, a razvoj, oblikovanje i isporuka razbiju se u inkrementalne dijelove koji predstavljaju djelomične funkcionalnosti. Što je presudno u definiranju prioriteta isporuke?

   Rj. Zahtjevi korisnika.

3. (1 bod) Scrum radni okvir koristi iterativni, inkrementalni pristup za optimizaciju predvidivosti i kontrole rizika, a sastoji se od Scrum timova i njihovih pridruženih uloga, događaja, artefakata i pravila. Opišite značajke razvojnog tima (engl. *development team*).

   Rj. Samoorganizirajući profesionalci koji isporučuju potencijalno isporučiv inkrement

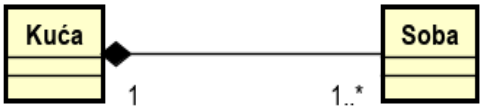
4. (1 bod) Koja je temeljna značajka agilnog modela (engl. *agile model*) razvoja programske potpore?

   Rj. razvoj se odvija iterativno i inkrementalno, ponavljaju se aktivnosti oblikovanja, implementacije i ispitivanja.

5. (1 bod) Navedite kojim ste sve dijagramima u projektnoj dokumentaciji prikazali arhitekturu sustava.

   Rj. dijagrami razreda, komponenti, razmještaja.

6. (1 bod) Objasnite kako je u implementacijskom smislu riješen odnos između razreda Kuća i Soba prikazanih na slici.



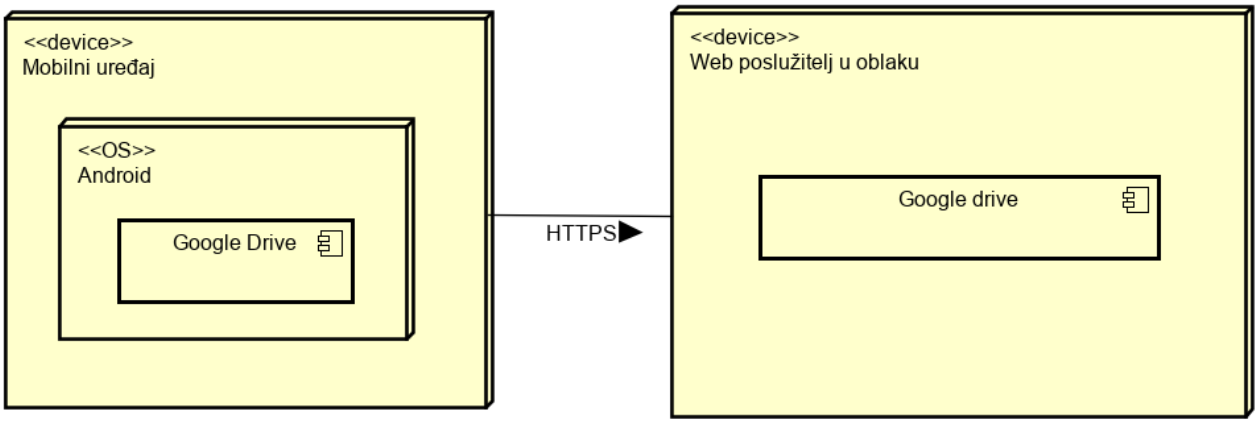
   Rj. Kompozicija: objekt razreda Kuća sadrži referencu na više objekata razreda Soba koje se stvaraju u konstruktoru razreda Kuća, a objekt razreda Soba sadrži referencu na jedan objekt razreda Kuća. Uništavanjem objekta razreda Kuća, uništavaju se i svi sadržani objekti razreda Soba.

7. (1 bod) Prilikom procesa razvoja dijagrama razreda, objasnite razliku između istraživačkog modela domene primjene, modela domene sustava i modela sustava.

Rj. Istraživački model domene sustava se crta s najmanje detalja i crta se s ciljem boljeg razumijevanja domene. Model domene sustava ima više detalja i modelira čitavu domenu, ali ne sadrži ostale razrede potrebne u izgradnji cjelovitog sustava. Model sustava uključuje sve razrede u aplikaciji i najdetaljniji je.

8. (2 boda) Nacrtajte specifikacijski dijagram razmještaja koji prikazuje komunikaciju mobilnog uređaja s operacijskim sustavom Android na kojem je mobilna aplikacija Google Drive s uslugom Google Drive u oblaku s web poslužiteljem. Mobilna aplikacija spaja se HTTPS protokolom na oblak.

Rj.



Napomena: priznaje se i stereotip <<service>> i <<cloud>> za web poslužitelj

9. (2 boda) Kod radnog okvira OCSF:

   a) (1 bod) Koliko najmanje dretvi treba biti prisutno na poslužiteljskoj strani, ako su na poslužitelj spojena tri klijenta? Napomena: zanemarite dretvu za komunikaciju s administratorom poslužitelja kojim se kontrolira rad poslužitelja.

   b) (1 bod) Koja je razlika između upravljačkih (`<<control>>`) metoda i metoda kopči (`<<hook>>`) kod razreda `AbstractServer`?

Rj.

a) 4 dretve – jedna koja osluškuje zahtjeve s klijenta i 3 za komunikaciju s klijentima.

b) Upravljačke metode imaju punu i dobro ispitanu implementaciju i ne mogu se nadjačati, dok metode kopči imaju trivijalnu implementaciju i služe da se redefiniraju (nadjačaju) u razredu koji nasljeđuje razred AbstractServer.

10. (1 bod) Kako je organiziran primjenski program u arhitekturi zasnovanoj na uslugama?

Rj. Uslužno usmjerena arhitektura organizira primjenski program (cjelovitu aplikaciju) kao kolekciju usluga koje međusobno komuniciraju uporabom dobro definiranih sučelja.

11. (1 bod) Radni okvir (engl. *framework*) kao skup integriranih komponenti omogućava ponovnu uporabu arhitekture za učestalo korišten dio programske potpore. Okviri su u osnovi nepotpuni, a korisnici okvira (programeri) ne smiju mijenjati kôd. Navedite barem dva načina razvoja programa iz radnih okvira.

Rj. Nasljeđivanjem komponenti, instanciranjem parametriziranih komponenti, razvojem funkcija koje nisu implementirane

12. (1 bod) Na primjeru trorazinske i troslojne arhitekture objasnite razliku između višerazinske arhitekture (engl. *n-tier*) i slojevite arhitekture (engl. *layered*).

Rj. Trorazinska arhitektura izvodi se na tri zasebna uređaja (okoline), dok se troslojna arhitektura može izvoditi na jednom, dva ili tri uređaja, a strukturno se sastoji od klijentskog sloja, sloja poslovne logike (logički sloj) i sloja baze podataka. Slojevita arhitektura odnosi se na strukturu modula, dakle statički pogled, višerazinska arhitektura (engl. tiered) odnosi na organizaciju u izvođenju (engl. run-time), dakle dinamički pogled.

13. (1 bod) U procesu razvoja programske potpore potrebno je pokazati da sustav odgovara specifikaciji i da zadovoljava zahtjeve kupca i korisnika. Navedite na koje pitanje daje odgovor generička aktivnost validacije u procesu programskog inženjerstva (engl. *validation*).

Rj: zadovoljava li sustav funkcijske zahtjeve

14. (1 bod) Funkciju koja kao ulaze prima dva cijela broja u rasponu [51, 100] potrebno je ispitati primjenom tehnike ispitivanja ekvivalentnih particija (engl. *equivalence partition testing*). Izračunajte broj potrebnih ispitnih slučajeva.

Rj.

rubne vrijednosti granice particija: 50,51, 100, 101

jedan element iz skupa valjanih ulaza: npr. 65

jedan element iz vrijednosti nevaljanih ulaza: 25, 444

Potrebno je ispitati kombinaciju svih podjela -> 7*7= 49 ispitnih slučajeva.

Napomena: priznaju se i rješenja s 5*5 = 25 ispitnih slučajeva

15. (1 bod) Navedite najmanje tri stvari koje treba ispitati prilikom ispitivanja komponente (engl. *unit testing*)?

Rj. sučelje, podaci (atributi), rubni uvjeti, iznimke, nezavisni putevi.

16. (2 boda) Za sljedeću funkciju:
    a) (1 bod) Nacrtajte graf tijeka programa.
    b) (1 bod) Odredite njezinu ciklomatsku složenost. Navedite formulu za njezin izračun.

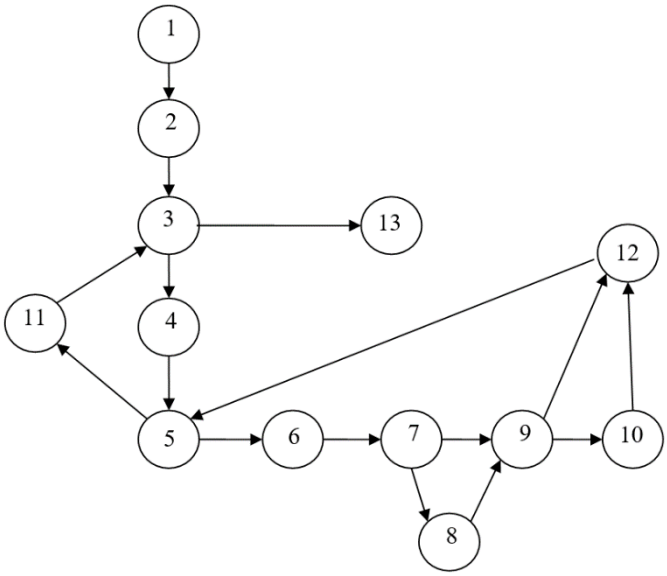
```
public double calcQIndex(double[] data, double sigma, int size) {
  double qIndex = 0.0;
  double factor;
  for (int i = 0; i < (size/8); i++) {
     for (int j = 0; j < (size/6); j++) {
          factor = Math.sqrt((i*j)*Math.PI);
        if (j != 0) {
              qIndex+=3.0*factor*sigma*(data[i*6+j]-data[i*6+j-1]));
        }
        if (i != 0) {
              qIndex+=1.5*factor*sigma*(data[i*6+j]-data[i*6+j-2]));
        }
     }
  }
  return qIndex;
```

```
}
```

Rj.

a)

```
double qIndex = 0.0;(1)
double factor;
for (int i = 0; (2) i < (size/8); (3) i++ (11)) {
   for (int j = 0; (4) j < (size/6); (5) j++ (12)) {
         factor = Math.sqrt((i*j)*Math.PI); (6)
        if (j != 0) { (7)
              qIndex+=3.0*factor*sigma*(data[i*6+j]-data[i*6+j-1])); (8)
        }
        if (i != 0) { (9)
              qIndex+=1.5*factor*sigma*(data[i*6+j]-data[i*6+j-2])); (10)
        }
   }
}
return qIndex; (13)
```



b) CV(G) = broj lukova - broj čvorova + 2*P (broj povezanih komponenti) = 16 - 13 + 2*1 = 5.

17. (1 bod) Trenutno se nalazimo na grani `develop`. Osim nje, postoji i grana `master`. Obje grane sadrže istu datoteku „`zaposlenici.txt`“ različitog sadržaja (prikazano na donjoj slici). Napišite kako će izgledati datoteka na trenutnoj grani nakon izvođenja naredbe „`git merge master`“.

zaposlenici.txt

```
1  Hrvoje
2  Igor
3  Nika
4  Ivan
```

grana master

zaposlenici.txt

```
1  Hrvoje
2  Igor
3  Nikolina
4  Ivo
```

grana develop

Rj.

zaposlenici.txt

```
1  Hrvoje
2  Igor
3  <<<<<<< HEAD
4  Nikolina
5  Ivo
6  =======
7  Nika
8  Ivan
9  >>>>>>> master
```

Napomena: ne priznaju se rješenja u kojima se navodi izgled datoteke **nakon** donošenja odluke prilikom konflikta.

17. (1 bod) Što mora biti zadovoljeno za interpretacije skupa formula **G**, tako da možemo tvrditi da vrijedi **G |= P** tj. da je formula **P** logička posljedica skupa **G**.

Rj. Svaka interpretacija koja za skup G daje istinitost mora i za P dati istinitost.

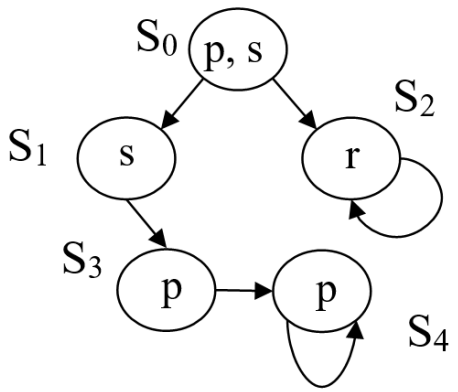
18. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:

" Postoji klijent koji je spojen s točno jednim poslužiteljem."

Rj.
K(x) = x je klijent
P(x) = x je poslužitelj
S(x,y) = x je spojen s y
=(x,y) = x je jednako y

$\exists x \exists y(K(x) \wedge P(y) \wedge S(x,y) \wedge !\exists z\ (P(z) \wedge S(x,z) \wedge !=(y,z))$

19. (1 bod) Za zadani model implementacije prikazan Kripke strukturom prema slici potrebno je navesti **skup** stanja koja zadovoljavaju formulu **EF p**.



Rj.

{S0, S1, S3, S4}

20. (1 bod) Prevedite rečenicu prirodnog jezika u formulu vremenske logike CTL:

„Postoji put od početnog stanja kojim konačno dolazimo u stanje od kojeg nadalje stalno vrijedi p.“

Rj.

a) EF AG p

21. Napomena: priznaje se i rješenje EF AX AG p

Problemski dio **– Perilica za rublje**

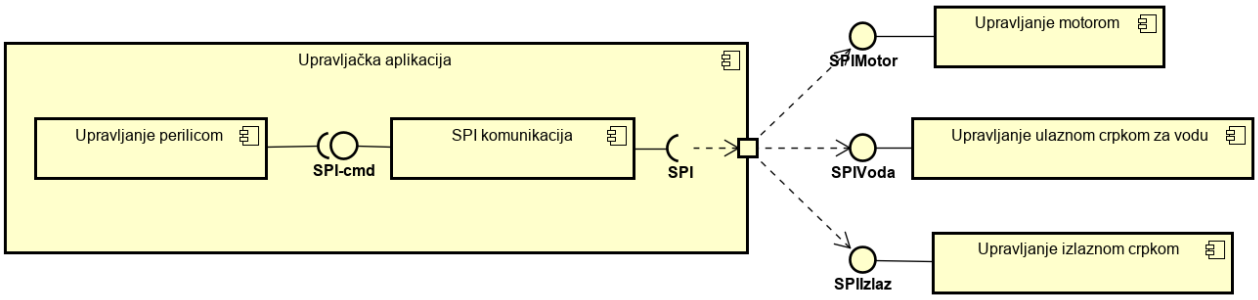
Glavni dijelovi jednostavne perilice za rublje su: korisničko sučelje, upravljačko računalo, motor koji okreće bubanj, ulazna crpka za vodu i izlazna crpka za vodu. Korisničko sučelje čine gumb ON/OFF, gumb START i zaslon na kojem se ispisuje oznaka trenutnog stanja perilice. Upravljačko računalo upravlja radom perilice i komunicira s ključnim dijelovima perilice protokolom SPI. Upravljački program je organiziran u dva modula: modul za upravljanje perilicom i modul za SPI komunikaciju za povezivanje sasvim ostalim dijelovima perilice. Modul za SPI komunikaciju implementira sučelje SPI-cmd koje koristi modul za upravljanje perilicom. Motor perilice, ulazna i izlazna crpka za vodu imaju integrirano upravljanje s komunikacijskim sučeljem SPI na koje se spaja komunikacijski modul upravljačke aplikacije glavnog računala.

Perilica za rublje je nakon montaže isključena. Uključuje se i isključuje pritiskom na ON/OFF gumb. Nakon što je uključena, upravljački program perilice na zaslonu ispisuje „00„ i čeka pokretanje pranja gumbom START. Nakon odabira pokretanja pranja, najprije se zaključavaju vrata od bubnja, a potom dinamički, ovisno o izmjerenoj masi robe u perilici određuje se potrebna faza pranja. Faza pretpranja neophodna je za rublje mase veće od 4kg. Faza deterdženta je prva faza pranja za rublje mase manje od 5 kg i na početku te faze generira se kratki zvučni signal. U fazi pretpranja na zaslonu se ispisuje „PP", a u fazi deterdženta se ispisuje „P". Daljnje faze pranja odvijaju se slijedno, a nakon faze deterdženta slijedi faza ispiranja. U fazi ispiranja se na zaslonu ispisuje „I". Kad je gotova faza ispiranja, slijedi faza omekšivala u kojoj se na zaslonu ispisuje „O". Kad je faza omekšivača gotova generira se kratki zvučni signal. Vrata perilice su zaključana još 30 sekundi nakon čega je moguće pokrenuti novo pranje. Ako u bilo kojem trenutku nestane struje ili korisnik isključi perilicu pritiskom ON/OFF gumba, perilica nakon ponovnog uključenja (povratka struje ili pritiska gumba) nastavlja s radom od početka faze u kojoj je stala. Ukoliko se prilikom rada perilice detektira da nema vode na ulazu, perilica se zaustavlja i na zaslonu se ispisuje „NV". Kad ponovno dođe voda na ulaz, perilica nastavlja s radom od početka faze u kojoj je stala.

21. (4 boda) Dijagram komponenti

Modelirajte UML dijagramom komponenti programske komponente perilice rublja i njihova sučelja.

Rj.

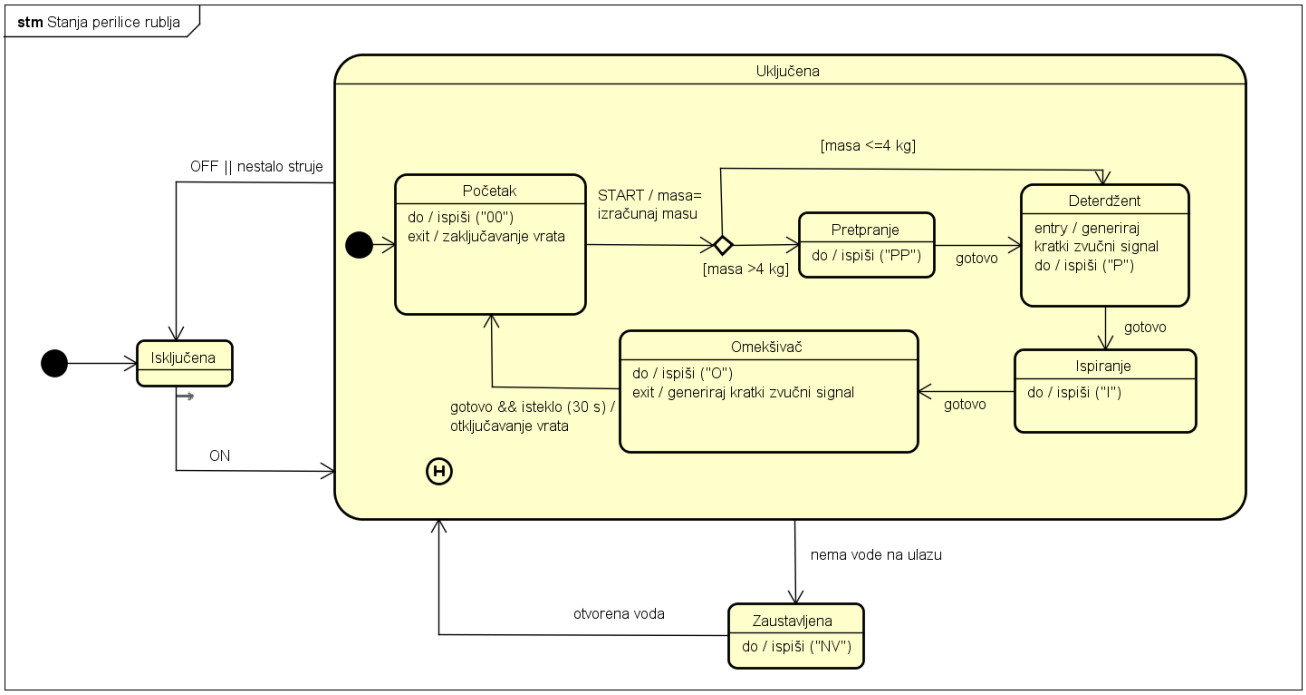


Napomene:

- Potrebno prepoznati sve komponente – priznaju se razne inačice „Upravljačke aplikacije", pa čak i „računalni modul" što nije potpuno ispravno rečeno
- U slučaju definiranja sučelja koje sadrži (skupno ili zasebno) zaslon, gumb on/off – taj dio je u okviru zadatka nepotreban, ali ukoliko postoji ne nosi nikakve bodove (niti pozitivne, niti negativne)
- Komponente moraju biti ispravno povezane (konektori) – u slučaju da nema ispravnih veza: - 0,5 boda
- Komponente moraju biti pravilno označene (na bilo koji ispravan način): -0,5 boda
- Veze moraju biti označene – ukoliko postoje, ali nisu označene: -0,25 boda

22. (4 boda) Dijagram stanja

UML dijagramom stanja modelirajte perilicu rublja. NAPOMENA: u fazama rada perilice nije potrebno modelirati uzimanje i izbacivanje vode koje je sastavni dio faze pranja.
Rj.



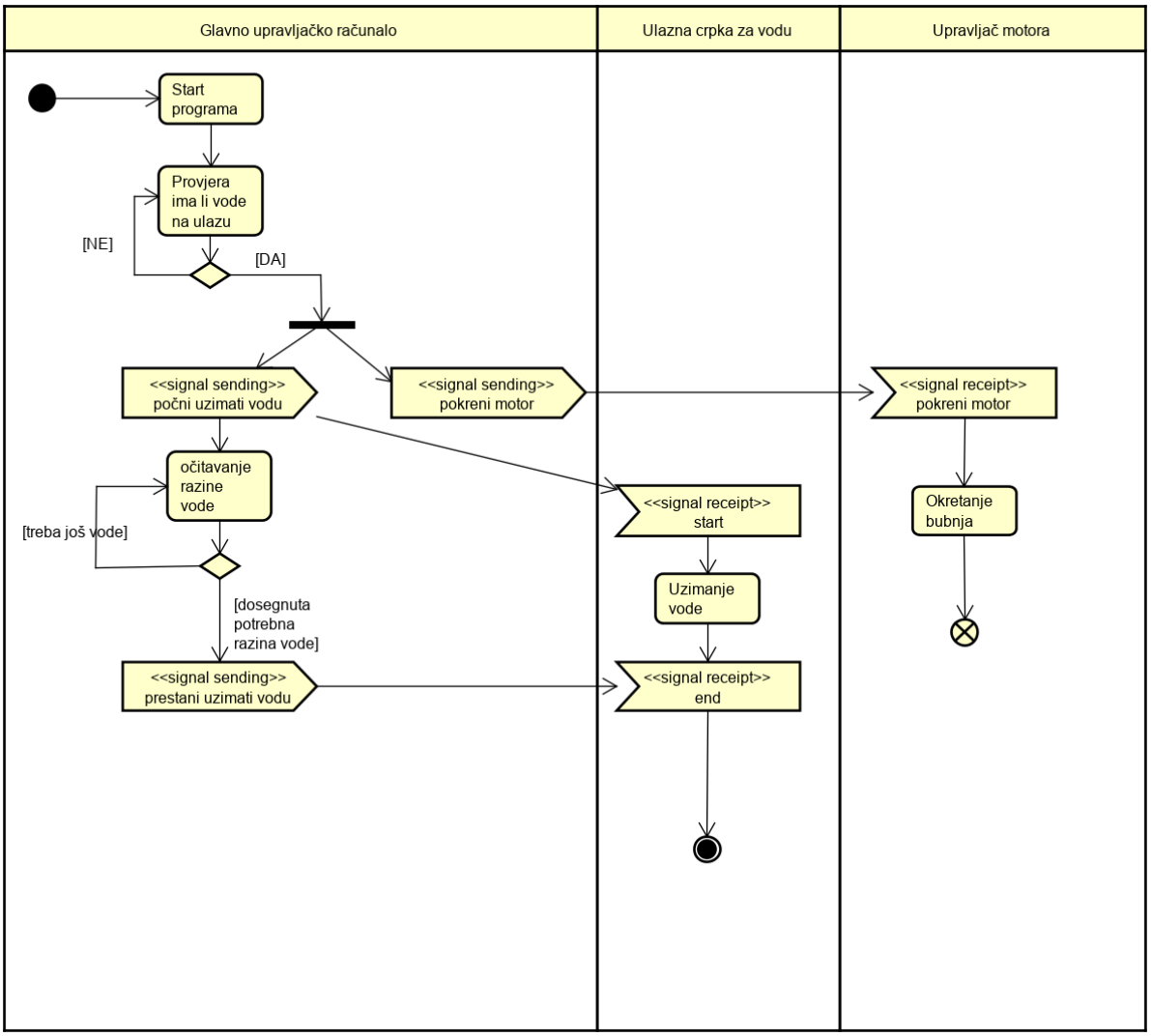
Semantika dinamičkog uvjeta i grananja 0,5 bod
Korištenje složenog stanja za smanjenje kompleksnosti dijagrama 0,5 bod
Korištenje povijesti za smanjenje kompleksnosti dijagrama 0,5 bod
Ispravna semantika prijelaza 0,5 bod
Ispravna notacija stanja (naziv, varijable stanja i akcije entry/do/exit) 1 bod
Korištena stanja i prijelazi ostvaruju traženo u zadatku 1 bod

23. (4 boda) Dijagram aktivnosti

Modelirajte UML dijagramom aktivnosti proces punjenja vode na početku svake faze pranja ako je zadano sljedeće:

Nakon pokretanja faze pranja upravljačko računalo perilice najprije provjerava ima li vode na ulazu u perilicu. Ako nema, čeka se sve dok ne dođe voda. Ako ima vode, istovremeno se ulaznoj crpki za vodu šalje signal da počne uzimati vodu, a upravljaču motora signal za pokretanje motora, nakon čega se krene okretati bubanj. Upravljačko računalo zatim očitava razinu vode sve dok nije dosegnuta potrebna razina vode i tada šalje signal crpki za vodu da prestane uzimati vodu te time završava proces punjenja vode.

Rj.



Neke od grešaka na kojima se gube bodovi:

- Pogrešni ili nepotrebni aktori: -0,25
- Fali početni ili završni čvor: -0,25
- Kod grananja, "čekanje na vodu" označeno kao akcija s loše modeliranom logikom prijelaza: -0,25 do -1
- Nisu korišteni signali: -0,5
- Neispravno korištenje čvora odluke: -0,5
- Netočno modelirano i ostvareno grananje: od -0,25 do -1
- Neispravno korištenje račvanja (engl. fork): -0,5
- Pogreške u redoslijedu izvođenja akcija i/ili fale neke akcije: od -0,25 do -2
- Neispravno korištenje čvora skupljanja (engl. join node): -0,5