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 A**

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

   Rj. Studija izvedivosti, izlučivanje zahtjeva , analiza i specifikacija zahtjeva, validacija zahtjeva, upravljanje zahtjevima.

2. (1 bod) Unificirani proces (engl. *Unified process*) naglašava kontinuiranu komunikaciju s korisnicima u svim fazama razvoja. Navedite kada korisnici mogu predložiti promjene i dopune?

   Rj. Na kraju svake iteracije.

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 ulogu Scrum vođe (engl. *Scrum Master*) u timu.

   Rj. Odgovoran za razumijevanje i primjenu teorije, prakse i pravila Scruma.

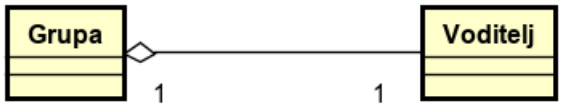
4. (1 bod) Koja je temeljna značajka vodopadnog modela (engl. *Waterfall model*) razvoja programske potpore?

   Rj. Prethodna faza treba se završiti prije prelaska na novu fazu.

5. (1 bod) Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili korisničke zahtjeve.

   Rj. Obrasci uporabe (use cases) s pripadajućim opisima scenarijima, sekv. dijagrami, lista nefunkcionalnih zahtjeva.

6. (1 bod) Objasnite kako je u implementacijskom smislu riješen odnos između razreda `Grupa` i `Voditelj`, prikazanih na slici.



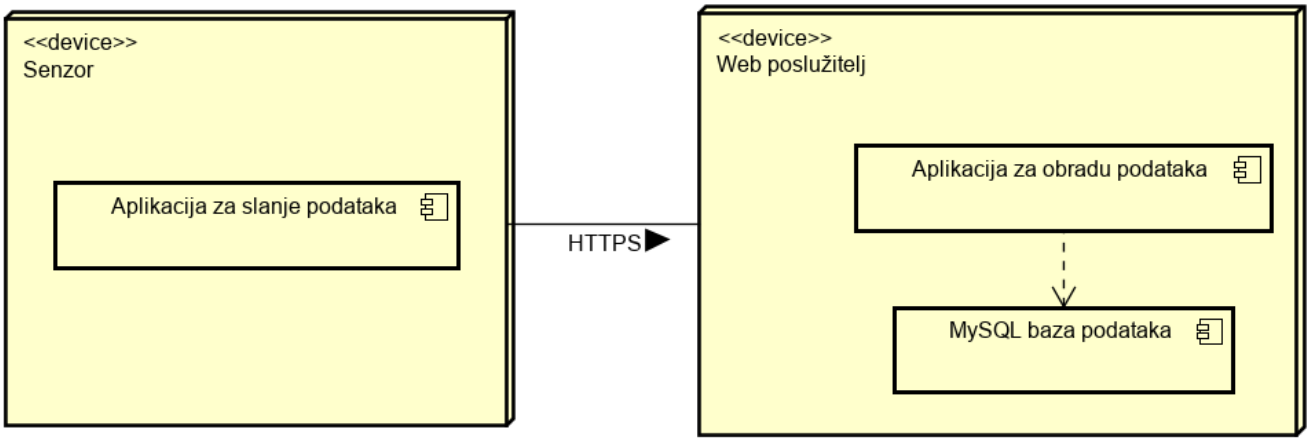
   Rj. Agragacija: objekt razreda Grupa sadrži referencu na jedan objekt razreda Voditelj, a objekt razreda Voditelj sadrži referencu na jedan objekt razreda Grupa. Životni ciklus objekta razreda Grupa i objekta razreda Voditelja nisu povezani.

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 senzora s web poslužiteljem. Aplikacija za slanje podataka na senzoru spaja se HTTPS protokolom na aplikaciju za obradu podataka koja se nalazi na poslužitelju. Aplikacija za obradu podataka ovisi o MySQL bazi podataka koja se nalazi na istom poslužitelju.

Rj.



Napomena: komponente mogu imati stereotip, kao što je <<artefact>>

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 `AbstractClient`?

Rj.

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

b) kontrolne 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 AbstractClient.

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) Oblikovni obrasci (engl. *design pattern*) su u praksi dokazano dobar način ponovne uporabe znanja o čestom problemu i načinu rješavanja te daju opis problema i osnovni predložak rješenja primjenjiv u različitim situacijama. Navedite tri osnovna tipa oblikovnih obrazaca.

Rj. Stvaralački (engl. *creational*), strukturni (engl. *structural*), ponašajni (engl. *behavioral*)

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 verifikacije (engl. *verification*) u procesu programskog inženjerstva.

Rj: Zadovoljava li sustav zahtjeve na ispravan način.

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

Rj.

ulazne vrijednosti -51..100: 152*152 = 23 104 ispitnih slučajeva.

15. (1 bod) Na koji se način provjerava ispravnost dobivenog i očekivanog izlaza u xUnit (JUnit, NUnit…) radnim okvirima za ispitivanje programske potpore?

Rj. Assert metodama.

16. (2 boda) Za sljedeću funkciju:

    a) (1 bod) Nacrtajte graf tijeka programa.

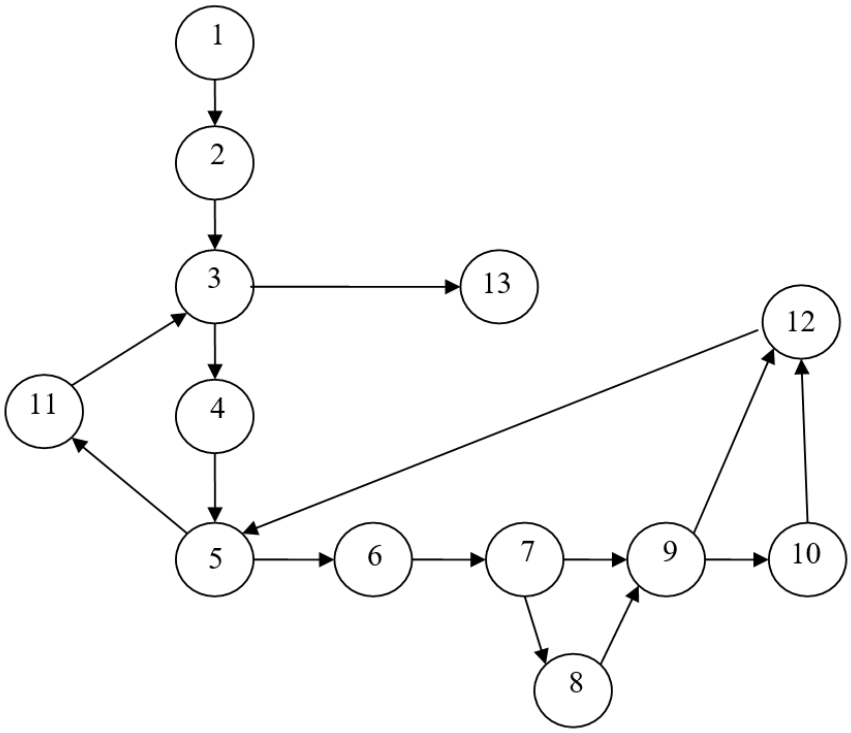
    b) (1 bod) Odredite njezinu ciklomatsku složenost. Navedite formulu za njezin izračun.

```
public Flux computeFlux(double[] q, double alpha, int sizex, int sizey) {
  Flux flux = new Flux(sizex, sizey);
  int index;
  for (int i = 0; i < sizex; i++) {
     for (int j = 0; j < sizey; j++) {
          index = getIndex(i,j);
          if (j != 0) {
                flux.addx(index, alpha*(q[index]-q[index-1])/2);
          }
          if (i != 0) {
                flux.addy(index, alpha*(q[index]-q[index-sizex])/2);
          }
     }
  }
  return flux;
}
```

Rj.

a)

```
Flux flux = new Flux(sizex, sizey); (1)
int index;
for (int i = 0;(2) i < sizex; (3) i++ (11)) {
   for (int j = 0; (4) j < sizey; (5) j++ (12)) {
        index = getIndex(i,j); (6)
        if (j != 0) { (7)
              flux.addx(index, alpha*(q[index]-q[index-1])/2); (8)
        }
        if (i != 0) { (9)
              flux.addy(index, alpha*(q[index]-q[index-sizex])/2); (10)
        }
   }
}
return flux; (13)
```



17. (1 bod) Trenutno se nalazimo na grani `master`. Osim nje, postoji i grana `develop`. 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 develop`“.

zaposlenici.txt

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

grana master

zaposlenici.txt

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

grana develop

Rj.

zaposlenici.txt

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

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

18. (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.

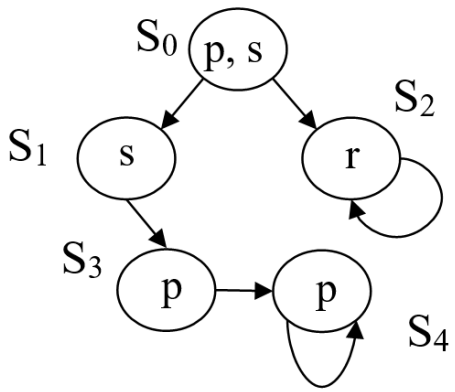
19. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:

"Svaki klijent spojen je s barem jednim poslužiteljem."

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

$\forall x\ (K(x) \Rightarrow \exists y(P(y) \wedge S(x,y))$

20. (1 bod) Za zadani model implementacije prikazan Kripke strukturom prema slici potrebno je navesti **skup** stanja koja zadovoljavaju formulu **EG p**.



Rj.

{S3, S4}

21. (1 bod) Prevedite rečenicu prirodnog jezika u formulu vremenske logike CTL:

„Na svakom putu od početnog stanja konačno dolazimo u stanje od kojeg nadalje stalno vrijedi p.“

Rj.

a) AF AG p

Napomena: priznaje se i rješenje AF AX AG p

Problemski dio **– Perilica za rublje**

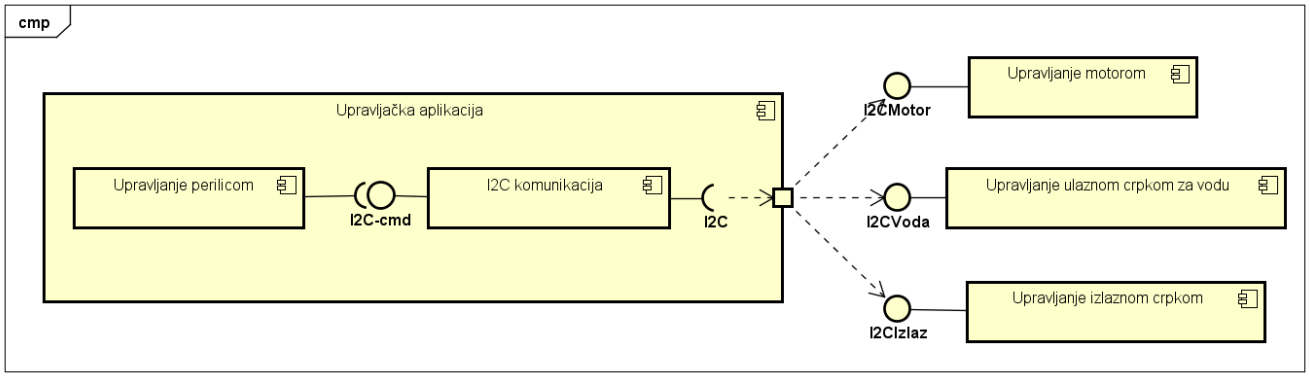
Glavni dijelovi jednostavne perilice za rublje su: korisničko sučelje, upravljačko računalo, motor koji okreće bubanj, ulazna crpka za vodu i izlazna crpka za vodu. Korisničko sučelje čine gumb ON/OFF, gumb START i zaslon na kojem se ispisuje oznaka trenutnog stanja perilice. Upravljačko računalo upravlja radom perilice i komunicira s ključnim dijelovima perilice protokolom I2C. Upravljački program je organiziran u dva modula: modul za upravljanje perilicom i modul za I2C komunikaciju za povezivanje sasvim ostalim dijelovima perilice. Modul za I2C komunikaciju implementira sučelje I2C-cmd koje koristi modul za upravljanje perilicom. Motor perilice, ulazna i izlazna crpka za vodu imaju integrirano upravljanje s komunikacijskim sučeljem I2C 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 „--„ 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 >=5 kg. 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 „F0", a u fazi deterdženta se ispisuje „F1". Daljnje faze pranja odvijaju se slijedno, a nakon faze deterdženta slijedi faza omekšivača. U fazi omekšivača se na zaslonu ispisuje „F2". Kad je gotova faza omekšivača, nakon čega slijedi faza ispiranja u kojoj se na zaslonu ispisuje „F3". Kad je faza ispiranja 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:

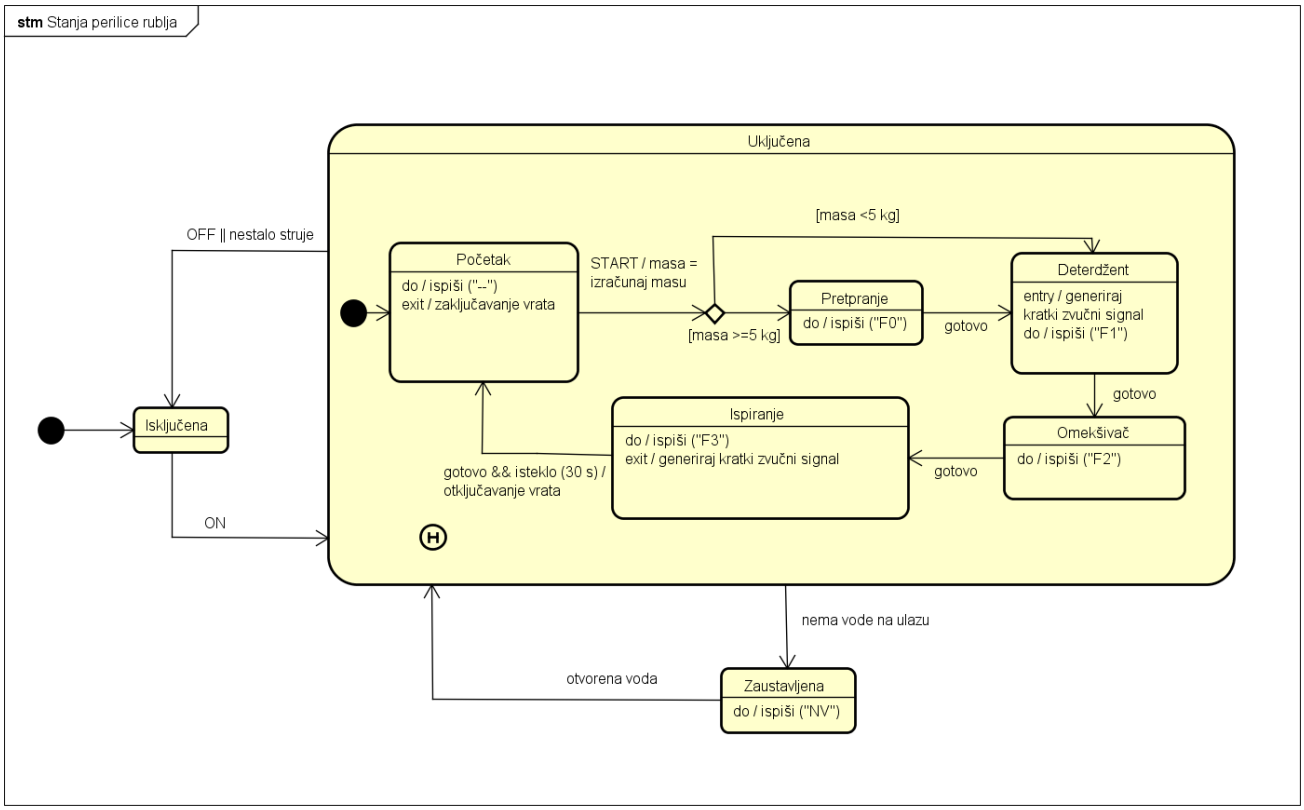
- Potrebno prepoznati sve komponente – priznaju se razne inačice „Upravljačke aplikacije", pa čak i „računalni modul" što nije potpuno ispravno rečeno
- U slučaju definiranja sučelja koje sadrži (skupno ili zasebno) zaslon, gumb on/off – taj dio je u okviru zadatka nepotreban, ali ukoliko postoji ne 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

- Nedostatak portova za komunikaciju: -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.



Kriteriji bodovanja:

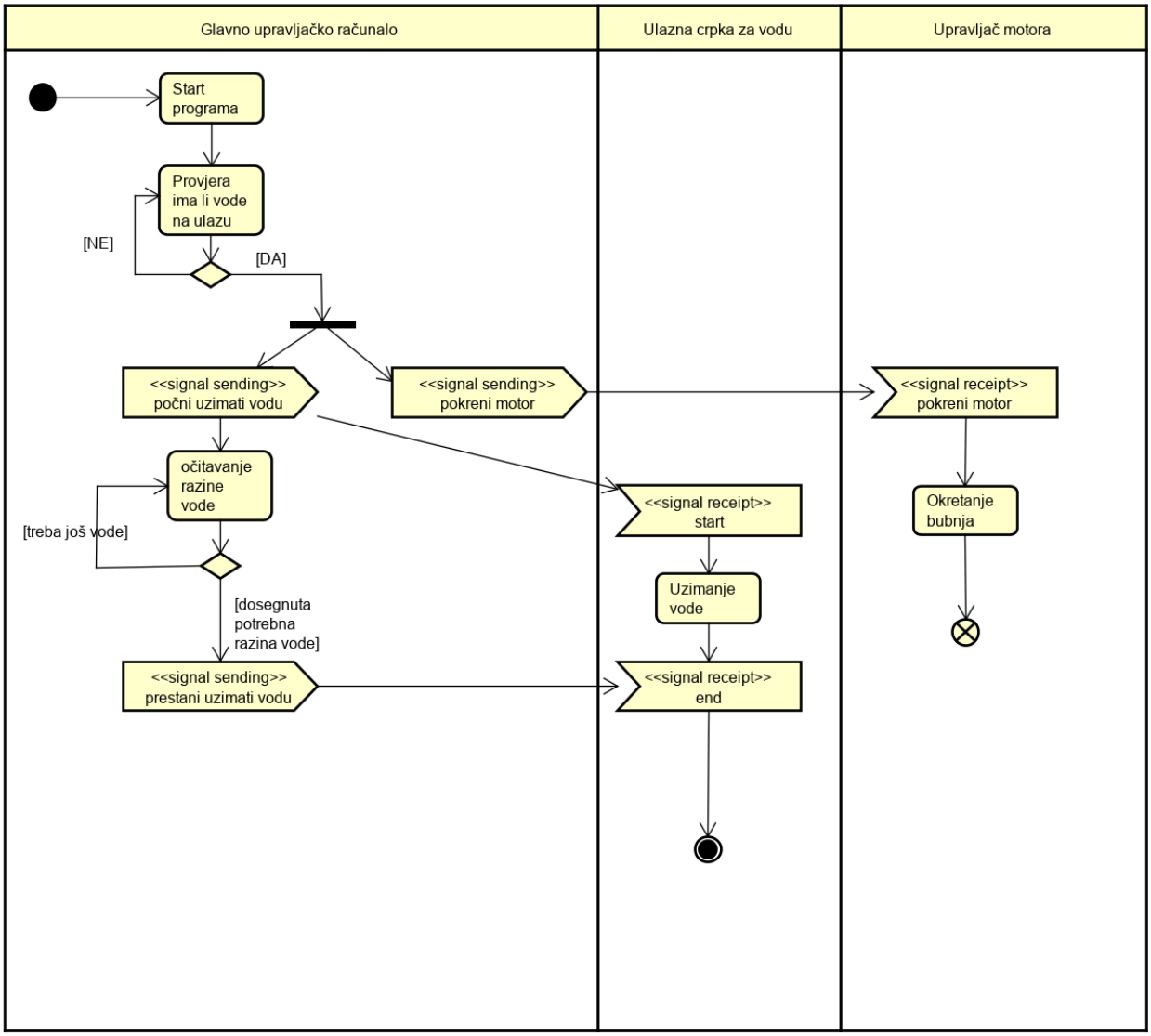
Semantika dinamičkog uvjeta i grananja 0,5bod
Korištenje složenog stanja za smanjenje kompleksnosti dijagrama 0,5bod
Korištenje povijesti za smanjenje kompleksnosti dijagrama 0,5
Ispravna semantika prijelaza 0,5
Ispravna notacija stanja (naziv, varijable stanja i akcije entry/do/exit) 1 bod
Korištena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