Pravila za izradu ispitnog zadatka iz MRKIRM I

Ilija Bašičević, ilija.basicevic@rt-rk.com

Miloš Pilipović, milos.pilipovic@rt-rk.com

Radovan Marković, radovan.markovic@rt-rk.com

Vladimir Kosjer, vladimir.kosjer@rt-rk.com

Elementi ocenjivanja

Boduju se sledeći elementi:
- koncept rešenja
- dokumentacija
- programski kod
- koncept testiranja
- realizovani testni slučajevi

Koncept rešenja

U izradi zadatka se očekuje da student prvo uradi dokumentaciju koja dovoljno jasno opisuje koncept rešenja, i da započne kodiranje tek nakon što asistent koji radi sa njim odobri urađenu dokumentaciju.
Time se izbegava situacija da nakon što je student uradio programski kod dobije ocenu da je koncept rešenja neodgovarajući i time manje bodova. Takođe se tim ubrzava proces izrade zadatka što je u interesu studenta.
Ukoliko je projekat deo veće celine u dokumentaciji a ponegde i u samom kodu naznačiti delove na kojima je student radio.

Izrada dokumentacije

Dokumentacija treba za većinu zadataka da sadrži minimalno SDL i MSC dijagrame.

Pravila za izradu SDL dijagrama:

1. Na jednoj strani prikazati grane kole proizilaze iz jednog stanja, odnosno za svako stanje izdvojiti novu stranicu dijagrama. (Osim u slučaju dijagrama automata trivijalne jednostavnosti.) Ukoliko iz jednog stanja proizilazi veči broj grana nego što je moguće pregledno prikazati na jednoj strani, tada se za to stanje koristi više strana. Pri tom nije

potrebno koristiti SDL konektore, već jednostavno svaka stranica započinje simbolom stanja.

2. SDL ima sledeću osnovnu strukturu za svaku granu:

stanje – prijem signala – slanje signala i-ili obrada – novo stanje
obrada se sastoji od simbola jednostavne obrade, poziva procedure i/ili grananja

Ovo pravilo jednostavno odslikava osnovni princip rada automata a to je da je reakcija na događaj (primljeni signal) slanje signala i/ili obrada, i da zatim automat prelazi u novo stanje.

3. Na dijagramu naznačiti početno stanje automata odgovarajućim SDL simbolom.
4. Ukoliko neka grana sadrži složeniju obradu, istu izdvojiti u SDL proceduru. Pri tome treba paziti da unutar procedure nema promene stanja, ni prijema signala. Drugim rečima dijagram procedure sadrži samo sledeće simbole: početak i kraj procedure, jednostavna obrada, poziv procedure, slanje signala i grananje.

Pravila za izradu MSC dijagrama

MSC dijagramima predstaviti normalno funkcionisanje sistema, kao i predviđene testne slučajeve. Ovi dijagrami će se razlikovati jer u slučaju kada se testira CPP Unitom, poruke automatu koji se testira ne šalje automat koji ih šalje u režimu normalnog funkcionisanja, već Message Factory automat.

U nekim zadacima se umesto SDL i MSC dijagrama mogu koristiti BDA (blok dijagram algoritma) dijagrami. To piše u postavci zadatka.
Poželjno je da se složenije obrade (kao što je parsiranje poruka složenijeg formata) prikažu u dijagramima SDL procedura ili BDA – a ne samo automat prelaza stanja.
Pored navedenih student može da ako smatra za potrebno uradi i dodatne dijagrame koji preciznije opisuju neki aspekt rešenja.

Pre predaje dokumentacije proveriti da li je ista u skladu sa navedenim pravilima.

Pravila za referenciranje literature

Ukoliko je u pitanju programska podrška ili rad (ispitni, diplomski, magistarski, doktorski) razvijen na fakultetu, dovoljno je navesti autora, katedru, godinu izrade.

Ukoliko je u pitanju knjiga navesti naziv, autora, godinu izdavanja i izdaavača.

Za dokumentaciju nekog programskog paketa, navesti naziv paketa, verziju i proizvođača.

Za reference na Internetu navesti naziv web prezentacije (na primer: wikipedia), naziv web stranice, i adresu.

Pravila za strukturiranje sistema automata

Sistem razmene poruka korišćenjem rutina jezgra realizuje komunikaciju unutar sistema automata, ali je posebno pitanje kako realizovati komunikaciju sistema automata sa okolinom. Uočavaju se dve tačke u kojima sistem automata komunicira sa okolinom: tastatura preko koje dobija podatke od lokalnog korisnika i WinSock sprega preko koje komunicira sa udaljenim računarom u mreži.
Studenti se upućuju da pogledaju implementaciju klasa NetFSM, TCPSystemSupport, UDPFSM, Client pošto se saradnjom ovih klasa realizuju klijent-server komunikacije na TCP/IP mreži. Tu se vidi kako se pokreće serverska nit koja prihvata konekciju na određenom portu, pokreće novu nit koja dalje opslužuje konekciju a sama nastavlja čekanje na pojavu sledećeg klijenta.
Očitavanje tastature se može realizovati na taj način da osnovna programska nit pokrene novu nit u kojoj se izvršava sistem nit, a sama nakon toga ulazi u petlju u kojoj prihvata znakove sa tastature i preko pokazivača na tzv. uslužni ili granični automat ubacuje poruke u sistem automata. Poruka se ubacuje tako da granični automat ima funkcije kojima sam sebi šalje određene poruke. Te funkcije poziva osnovna nit preko pokazivača, a poruke ulaze u granični automat kao da su stigle iz nekog drugog automata.
Isti princip – sprega za slanje poruka samom sebi se koristi i u Client klasi za poruke pristigle sa mreže.

Pravila za pisanje programske podrške ispitnog rada iz MRKiRM I

Imenovanje:

- U imenovanju promenljivih i funkcija koristiti pristup primenjen u izradi FSM biblioteke. Nazivi metoda počinju velikim slovom. Ukoliko se naziv metode ili atributa sastoji od više reči, spajanje reči se vrši tako što svaka nova reč počinje velikim slovom.

Primer: uint16 NumOfTimers;
void SetLeftAutomate(uint8 automate)

- Ne koristiti donju crtu (underscore) za spajanje reči u nazivima.
- Ne koristiti mađarsku notaciju.
- Nazivi treba da odražavaju namenu imenovane promenljive ili funkcije.

Primer: void SetInitialState()

- Izbegavati preduge nazive – preporučuje se da naziv bude kraći od 20 znakova.

Modularnost:

- Tela funkcija koje obrađuju događaje u pojedinim stanjima (event handler funkcije) treba da budu kratka, radi preglednosti. Sve duže obrade izdvojiti u posebne funkcije. Ukoliko su u programu prisutne funkcije duže od 100 linija, sa velikom verovatnoćom će se za to skidati bodovi.
- Definiciju klase izdvojiti u posebnu .cpp datoteku.

Formatiranje i komentari:

- Uvlačenje blokova realizovati u dubinu od 4 znaka.
- Komentarisati bitne elemente programskog koda koji nisu očiti iz samog koda. Očite osobine koda ne treba komentarisati.

Programski kod koji se predaje kao rezultat rada treba da bude očišćen od zakomentarisanih delova koda.

Koncept testiranja

Testiranje je bitan deo izrade zadatka i tako se boduje.

Očekuje se da student identifikuje i opiše korišćenjem MSC dijagrama ili makar tekstualno 4-6 testnih slučajeva.

Od studenta se očekuje da minimalno u kod ugradi mehanizam zapisivanja događaja koji omogućuje kasniju analizu. U tom slučaju će student tokom odbrane pokrenuti testne slučajeve i nakon završetka rada programa prikazati fajlove sa zapisima događaja.

Sledeći nivo je automatsko testiranje, koje donosi više bodova.

Savetuje se korišćenje CPP Unit biblioteke za testiranje, kao u primeru sa vežbi.
Ukoliko student ima ideju kako da testiranje izvede na drugi način rešenje će se prihvatiti pod uslovom da:

1. Rešenje omogućuje kontrolu nad izvršenjem programa korak po korak
2. Omogućuje automatsku proveru uspeha izvršenja testnog slučaja

U zadacima koji koriste Winsock, a ne koriste FSM biblioteku, savetuje se studentima da programsko rešenje koncipiraju tako da njihov automat predstavlja jezgro aplikacije. Jezgro koristi funkcije Winsock-a preko modula sprege, a taj modul u režimu testiranja preusmerava pozive funkcijama prijema i slanja paketa na *messagebox* koji je dostupan iz klase TestCase odnosno MessageFactory.
Režim funkcionisanja kontrolisati preprocesorskim direktivama (#define, #ifdef, #endif).

Napomena: klasa koju student razvija treba da ima naziv naziv različit od „TestCase“. Inače će doći do grešaka u prevođenju.

Vazna napomena: Programska podrška rešenja zadatka sadrži automate koji se svrstavaju u dve grupe:

- jezgro – automati koji realizuju logiku aplikacije
- uslužni – najčešće realizuju komunikacione usluge (sprega ka Winsock bibliteci itd)

Programska podrška projekta sadrži automate koji se svrstavaju u dve grupe:

- jezgro
- automati namenjeni za testiranje (message factory itd)

Jezgro u oba projekta mora biti isto!
Odstupanje od ovog pravila dovodi do značajno manjeg broja bodova.

Predaja zadatka

Nakon što je uspešno odbranio zadatak, student treba da u zip arhivu spakuje fajl sa dokumentacijom i ceo Visual C/C++ projekat sa kodom rešenja. Ako je testiranje rađeno u zasebnom projektu, spakovati oba projekta. Onaj deo koda koji nije u arhivi se neće bodovati, tako da na primer ako student ne uključi projekat sa testovima, smatraće se da testovi nisu urađeni.
Pri tome treba izbrisati sadržaj Debug direktorijuma.
Arhivu nazvati po obrascu: PrezimeStudenta_brojIndeksa.