

FORMALNA SPECIFIKACIJA PROTOKOLA, I deo

- ◆SDL (Specification and Description Language)
- ◆MSC (Message Sequence Chart)

POJAM PROTOKOLA

- ◆ Neformalna definicija: Protokol je skup dogovora (konvencija) i pravila o njihovoj primeni, koja definišu način komunikacije posmatranog entiteta sa njegovim okruženjem.
- ◆ Ono što blok dijagram algoritma predstavlja za proces obrade, to protokol predstavlja za komunikacioni proces.

KOMUNIKACIONI PROTOKOL ČINE:

- ◆ Skup poruka sa definisanim formatom (oblikom).
- ◆ Opis obrade pojedinih poruka – skup reakcija komunikacionog procesa.
- ◆ Način obrade grešaka – reakcija na nezvaredne događaje.

NEFORMALNA SPECIFIKACIJA

- ◆ Se najčešće sreće kao tekstualno-grafički opis karakterističnih scenarija komunikacije.
- ◆ Ona obično ne govori ništa o redosledu zadataka.
- ◆ Najčešće nije potpuna. I to najčešće nedostaje specifikacija vremenskih kontrola (tajmera).

TIPIČAN PRIMER NEFORMALNE SPECIFIKACIJE ZADATAKA

- ◆ Majka polazi na posao i kaže ćerki:
 - 1) Nemoj zaboraviti da uradiš domaći.
 - 2) Kad ogladniš doručkuj.
 - 3) Pre nego što kreneš u školu izbaci đubre.

FORMALNA SPECIFIKACIJA

- ◆ Treba da specificira sva stanja i prelaze između stanja, uključujući vremenske okvire, na jednoznačan način.
- ◆ ITU-T je standardizovao jezike za specifikaciju (SDL i MSC) i testiranje i verifikaciju (TTCN) komunikacionih protokola.
- ◆ Danas se koristi i objektni pristup – UML (Unified Modeling Language).

SDL (*Specification and Description Language*)

- ◆ SDL je standardni jezik za specificiranje i opis sistema, pre svega sistema koji rade u realnom vremenu
- ◆ Ovde spadaju i telekomunikacioni sistemi.
- ◆ Osnovni skup pravila dat je u preporuci Z.100e. Dodatna objašnjenja sadržana su u nekoliko narednih preporuka Z.100d1e, Z.100nce, Z.100nfe, Z.100p1e i Z.100s1e.

OSOBI NE SDL JEZIKA

- ◆ jednostavan za učenje,
- ◆ da se jednostavno može proširiti pojavom novih zahteva i
- ◆ da može da podrži više metodologija specificiranja sistema.

DVE PREZENTACIJE SDL-A

- ◆ grafička (SDL-GR) i
- ◆ programska (SDL-PR).

Aplikacije SDL-a

- ◆ obrada poziva u komutacionim sistemima,
- ◆ nadzor i reakcija na pojavu grešaka u telekomunikacionim sistemima,
- ◆ upravljanje sistemima,
- ◆ protokoli za prenos podataka,
- ◆ telekomunikacione usluge.

Osnove SDL jezika

- ◆ SDL je definisan skupom specijalnih znakova i pravilima njihovog korišćenja.
- ◆ U slučaju programske prezentacije koristi se skup rezervisanih reči, dok se u slučaju grafičke prezentacije koristi skup rezervisanih simbola.
- ◆ Za prikaz podataka u sistemu kod obe predstave koristi se isti skup rezervisanih reči.

OPIS SISTEMA

- ◆ Sistem se opisuje skupom međusobno povezanih blokova.
- ◆ Kanali definišu komunikacione puteve kojima blokovi komuniciraju međusobno ili sa okruženjem.
- ◆ U svakom bloku definiše se jedan ili više procesa.
- ◆ Procesi međusobno komuniciraju pomoću signala.

OPIS SISTEMA - nastavak

- ◆ Svakom kanalu se često pridružuje FIFO red u koji se smeštaju signali namenjeni tom kanalu.
- ◆ Procesi se definišu automatima s konačnim brojem stanja.

Jednostavan primer grafičke i programske prezentacije SDL jezika

- ◆ Na primeru jednostavne igre Daemon Game prikazuju se obe prezentacije SDL jezika.
- ◆ Igra se sastoji u sledećem: vremenskom kontrolom u sistemu menja se stanje tekuće promenljive sa vrednosti Win na Lose i obratno.
- ◆ Igrač upitima proverava u proizvoljnim trenucima vremena stanje tekuće promenljive te ukoliko na kraju igre skupi više Win stanja pobedio je, u protivnom je izgubio.

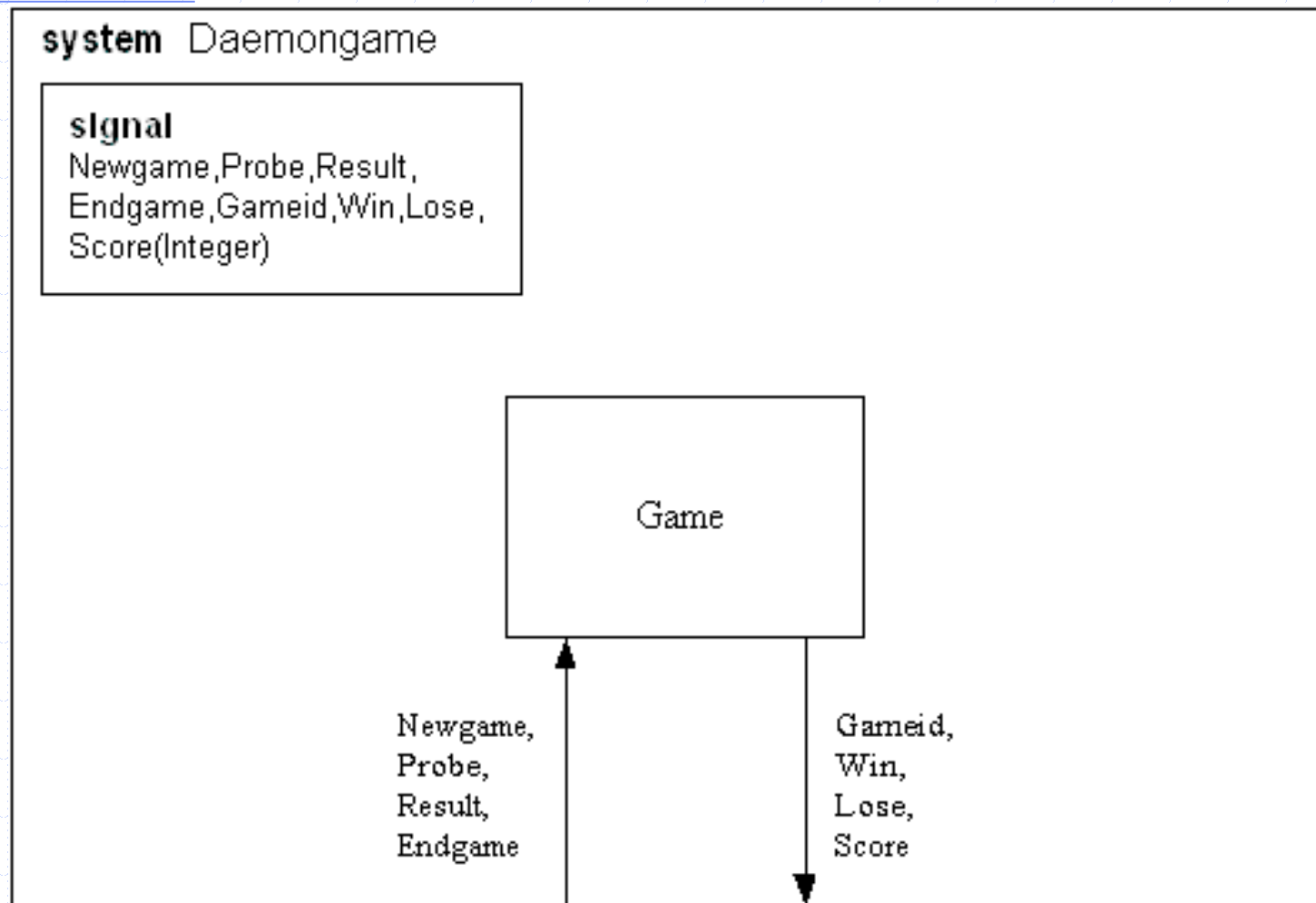
Ulazni signali su:

- ◆ Newgame – signal kojim igrač započinje igru,
- ◆ Probe – signal kojim igrač proverava stanje tekuće promenljive,
- ◆ Result – signal kojim igrač završava igru,
- ◆ Endgame – signal kojim se izlazi iz igre.

Izlazni signali su:

- ◆ Gameid – broj igre koja je u toku,
- ◆ Win – trenutno stanje igre Win,
- ◆ Lose – trenutno stanje igre Lose,
- ◆ Score – krajni rezultat.

Grafička prezentacija



Programska prezentacija

system Daemongame

signal Newgame, Probe, Result, Endgame, Gameid, Win, Lose, Score(Integer);

channel Gameserver.in

from env to Game

with Newgame, Probe, Result, Endgame;

endchannel Gameserver.in;

channel Gameserver.out

from Game **to** env

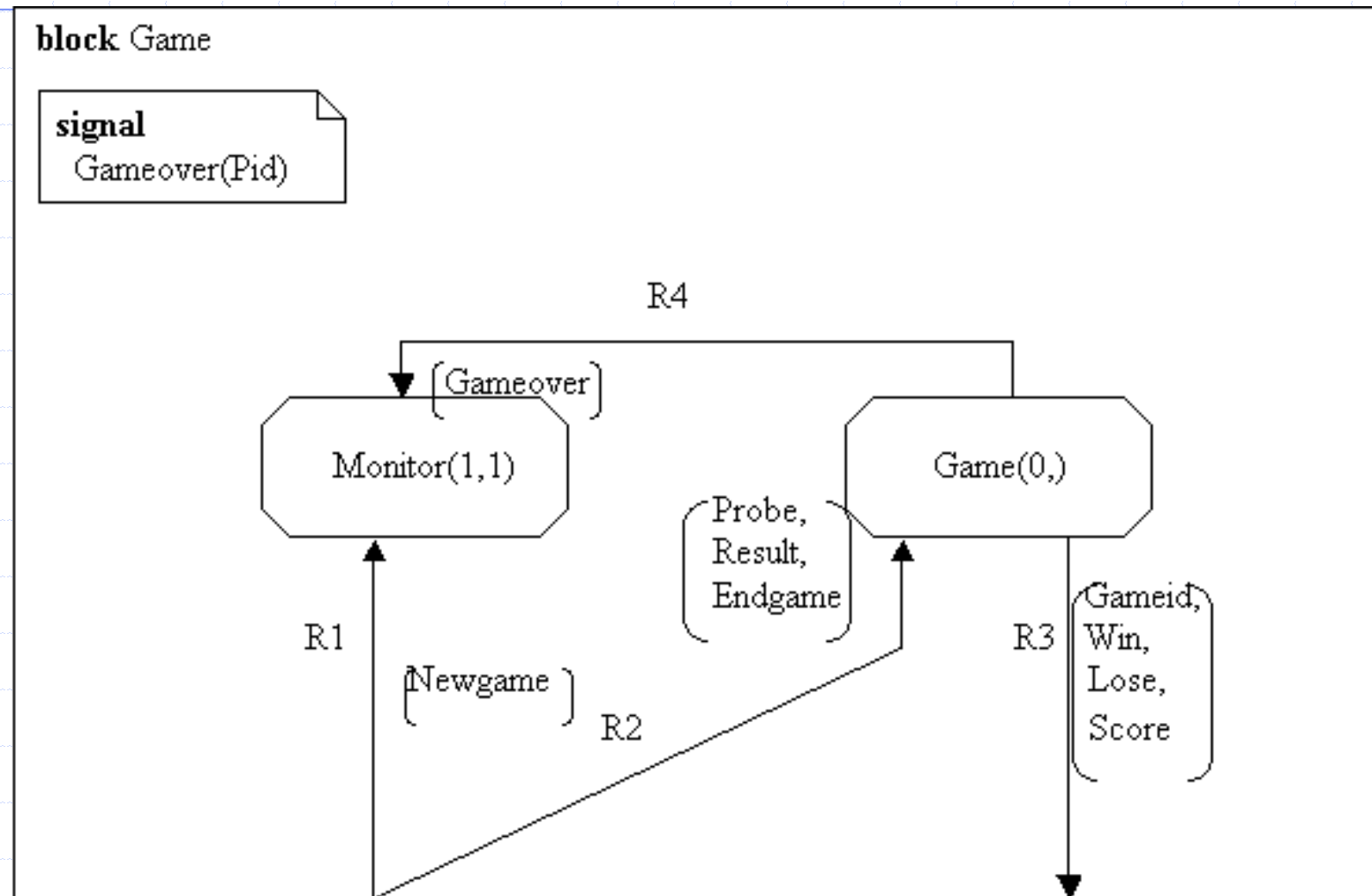
with Gameid, Win, Lose, Score;

endchannel Gameserver.out;

block Game referenced;

endsystem Daemongame;

Grafička predstava bloka Game



Programska predstava bloka Game

```
block Game;  
  signal Gameover(Pid);  
  connect Gameserver.in and R1, R2;  
  connect Gameserver.out and R3;  
  signalroute R1 from env to Monitor with Newgame;  
  signalroute R2 from env to Game with Probe,Result,Endgame;  
  signalroute R3 from Game to env with  
    Gameid,Win,Lose,Score;  
  signalroute R4 from Game to Monitor with Gameover;  
  process Monitor(1,1) referenced;  
  process Game(0,) referenced;  
endblock Game;
```

Definisanje signala

- ◆ Definisanje signala obavlja se ključnom rečju **signal** iza koje sledi ime signala Gameover.
- ◆ Nakon imena može da se navede lista veličina koje čine ovaj signal, u datom slučaju to je vrednost Pid.
- ◆ Pid predstavlja broj kojim se na jedinstven način identifikuju procesi u sistemu.

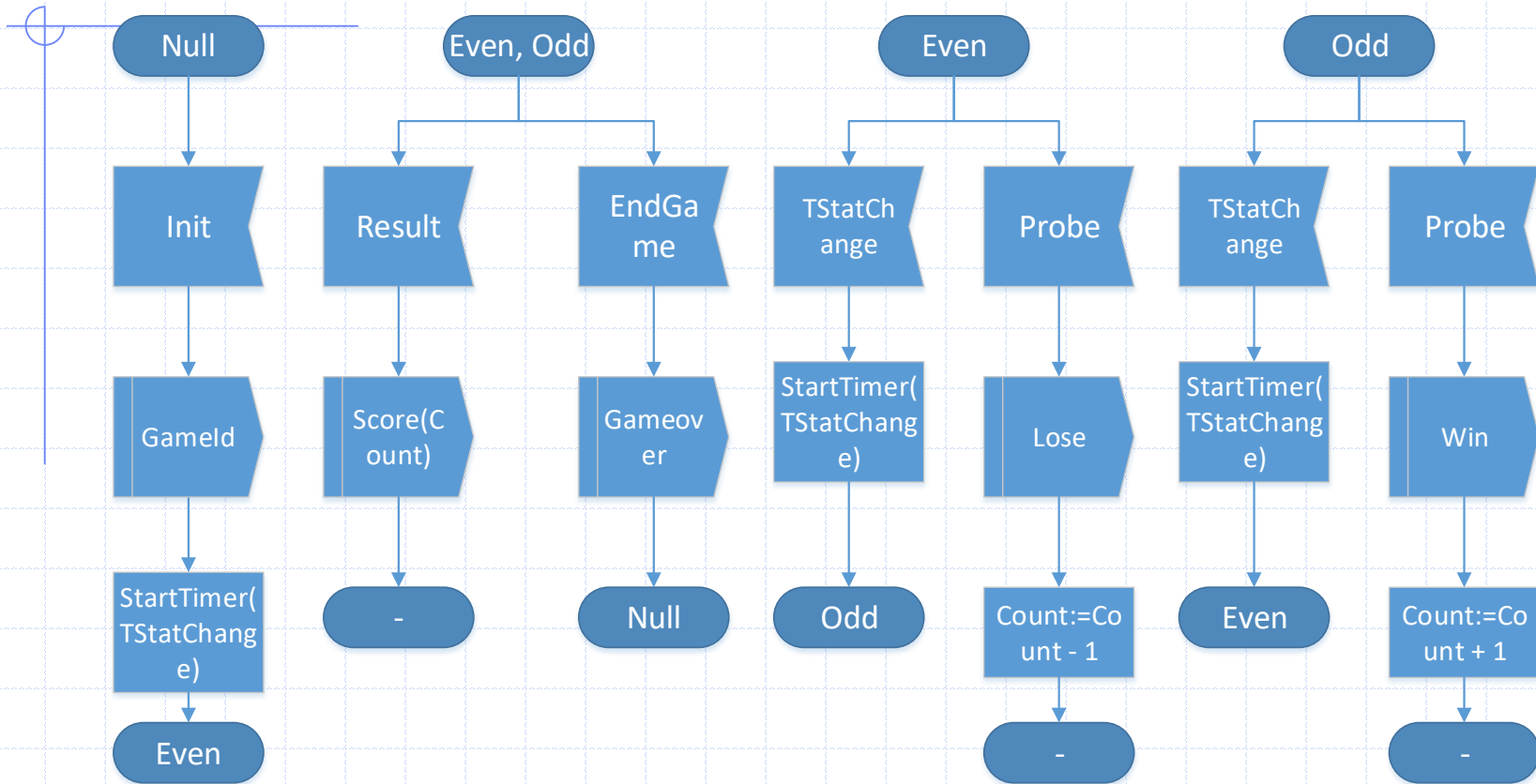
Definisanje procesa

- ◆ Definisanjem procesa sem imena procesa može se postaviti i broj instanci tog procesa koje se mogu pojaviti u sistemu.
- ◆ Broj instanci se predstavlja parom (inicijalan broj instanci, maximalan broj instanci), pri čemu maksimalan broj ne mora biti definisan.

Procesi u bloku Game

- ◆ U okviru bloka Game definisana su dva procesa, Monitor i Game.
- ◆ Procesom Monitor predstavljen je rad sa tastaturom.
- ◆ Zadržaćemo se na definisanju procesa Game.

Grafička predstava procesa Game



Programska predstava procesa Game (1/2)

```
process Game(0,); fpar player Pid;  
  dcl count Integer := 0; /* counter to keep track of score */  
  start;  
    output Gameid to player;  
    nextstate even;  
  state even;  
    input TStatChange;  
    nextstate odd;  
  input Probe;  
    output Lose to player;  
    task count:=count-1;  
    nextstate -;
```

Programska predstava procesa Game (2/2)

```
state odd;  
  input Probe;  
    output Win to player;  
    task count:=count+1;  
    nextstate -;  
  input TStatChange;  
    nextstate even;  
state even,odd;  
  input Result;  
    output Score(count) to player;  
    nextstate -;  
  input Endgame;  
    output Gameover(player);  
  stop;  
endprocess Game;
```

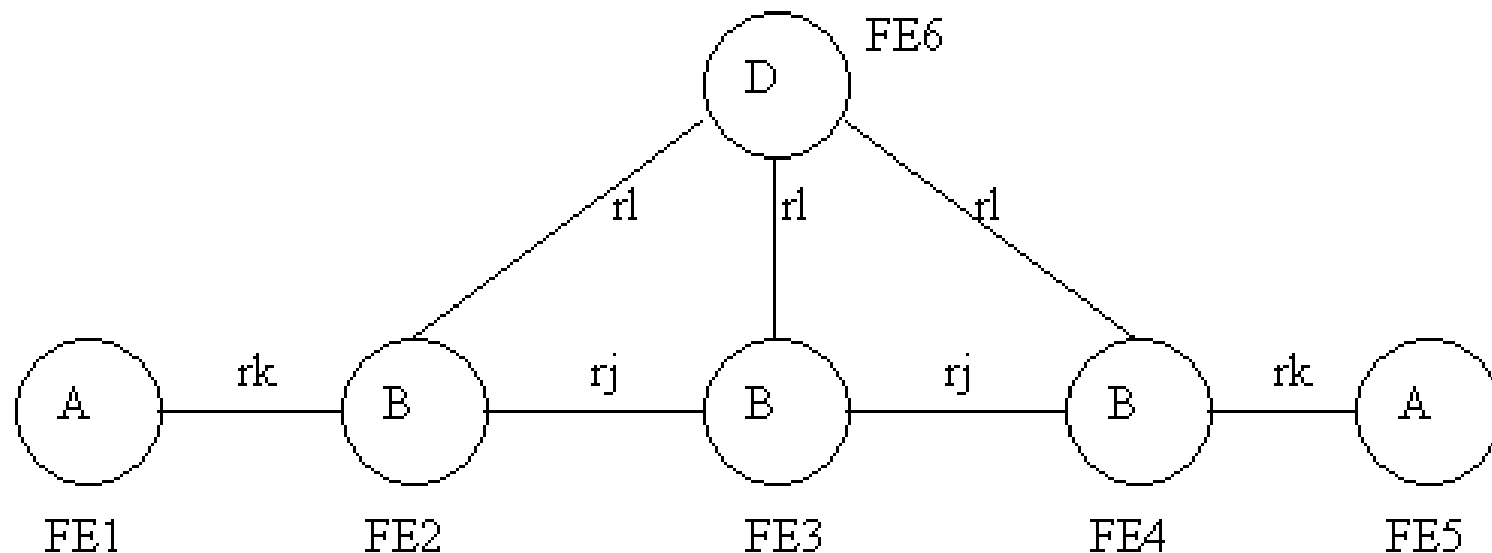
Definisanje formalnih parametara

- ◆ U prikazanom procesu može se videti i ova konstrukcija **fpar player Pid**, koja predstavlja definisanje formalnog parametra **player** koji se identifikuje vrednošću Pid.
- ◆ U trenutku kada igra počne u sistemu će biti pokrenuta nova instanca kojoj će biti dodeljen njen Pid broj.

Primer: Upravljanje telefonskim pozivima

- ◆ Sledeći primer prikazuje korišćenje SDL jezika za definisanje sistema upravljanja pozivima u telefonskom saobraćaju.
- ◆ Opis ovog sistema dat je u posebnoj preporuci Q.71 .
- ◆ Sastoji se iz skupa međusobno povezanih funkcionalnih entiteta.

Funkcionalni model ovog sistema



gde su:

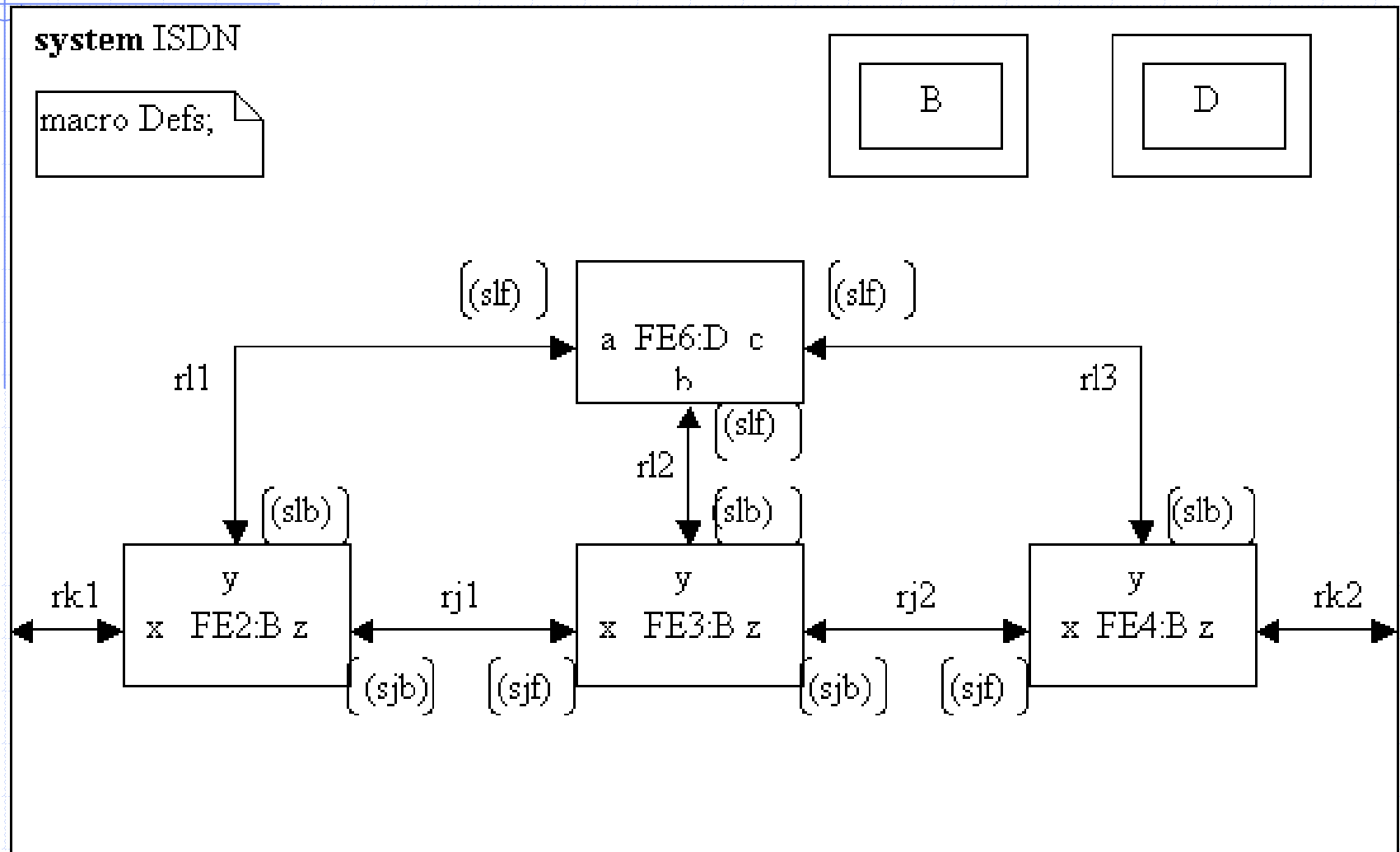
A,B,D – tip funkcionalnog entiteta

FE1,FE2... – ime funkcionalnog entiteta

rk,rj,rl – veze između tipova funkcionalnih entiteta

SDL opis sistema

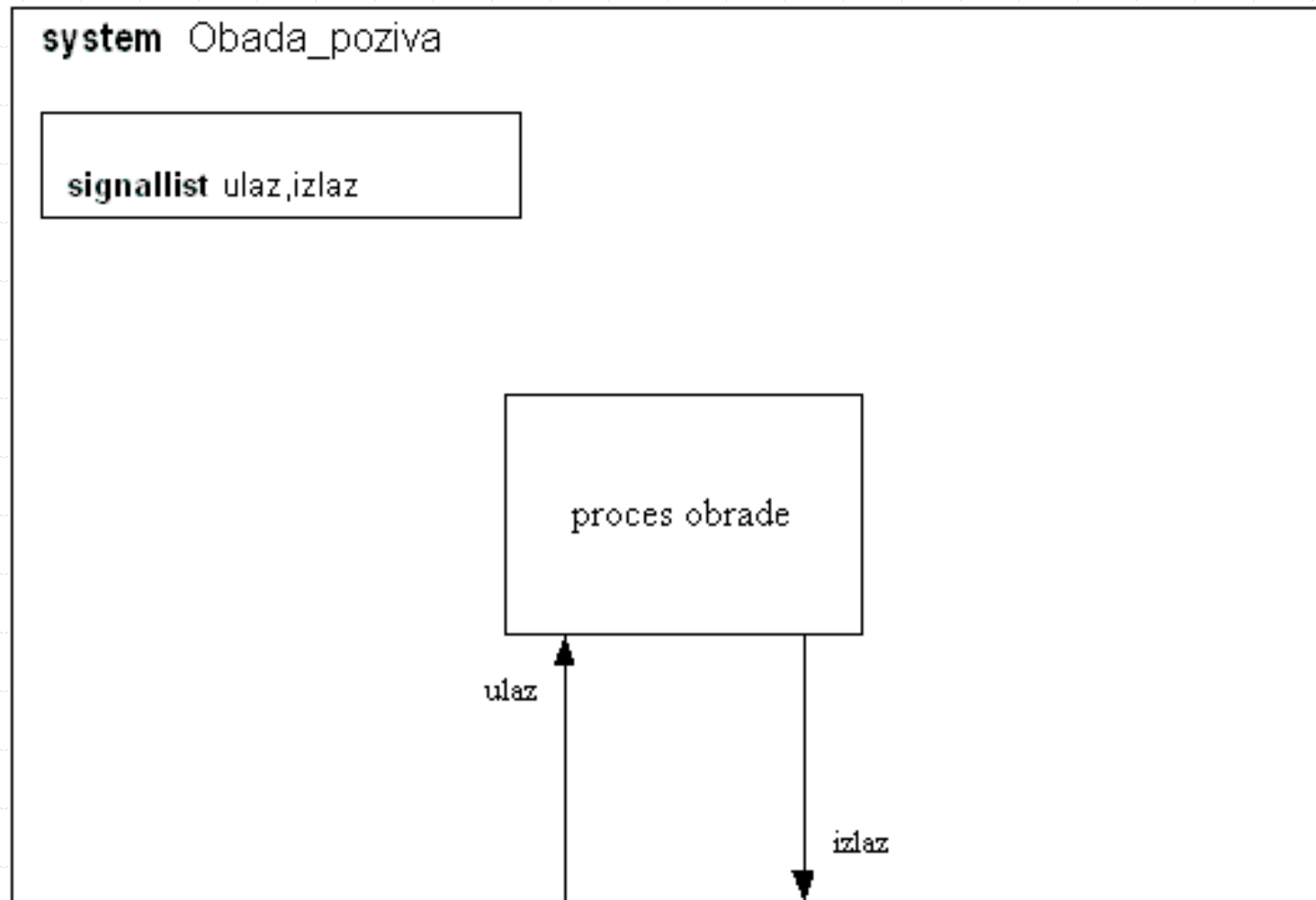
(Sadržaj blokova kojima su opisani funkcionalni entiteti predstavljaju procese.)



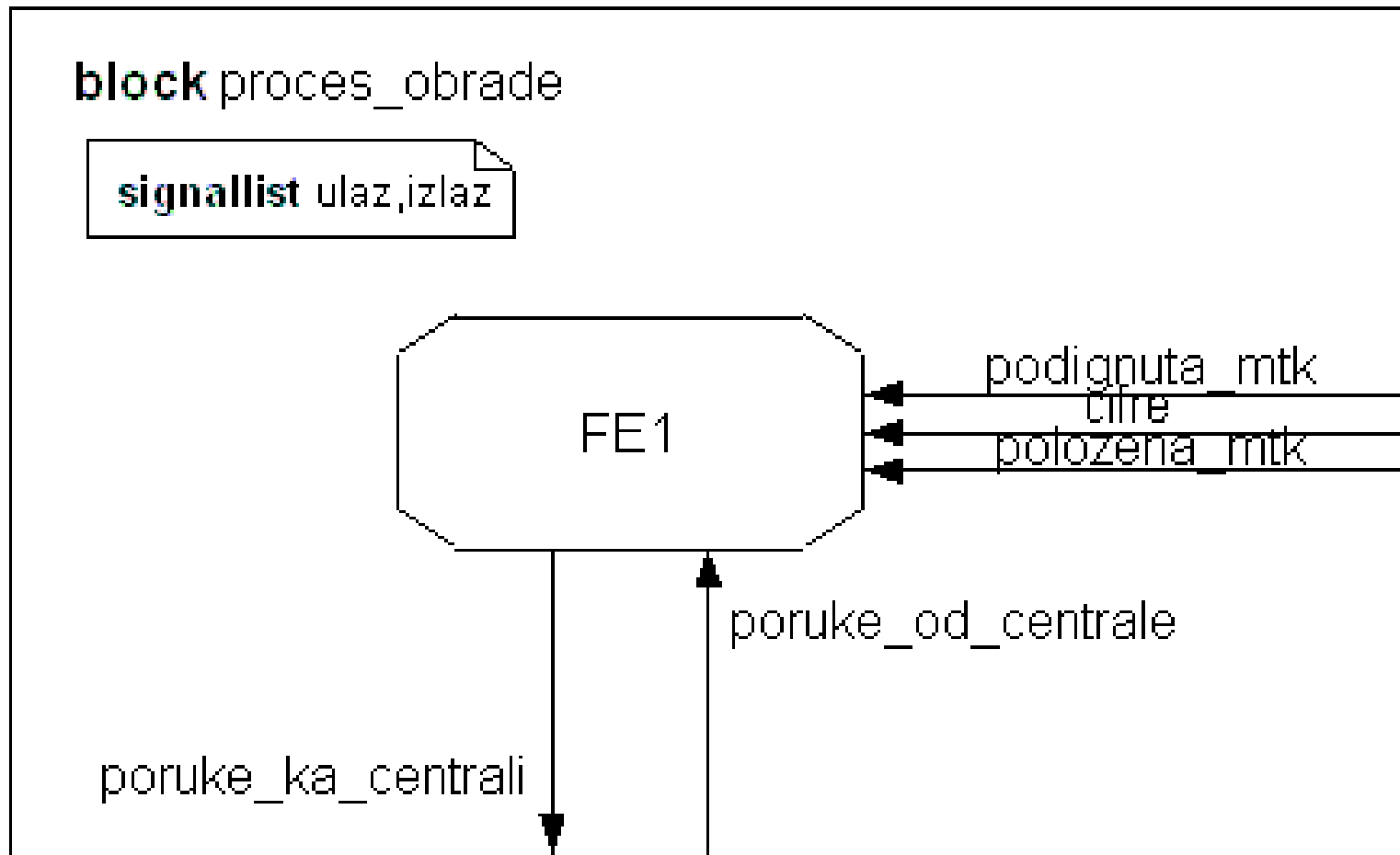
Primer: FE1

- ◆ Primena SDL na primeru FE1 prikazana je na naredne tri slike.
- ◆ Prva slika prikazuje izgled sistema FE1, druga sadržaj bloka proces_obrade, dok treća slika prikazuje uprošćen SDL dijagram samog FE1 automata.
- ◆ Vidi se da FE1 prima signale od centrale i od rutine koja nadgleda fizičko stanje linije. Od FE1 automata signali se prosleđuju ka centrali.

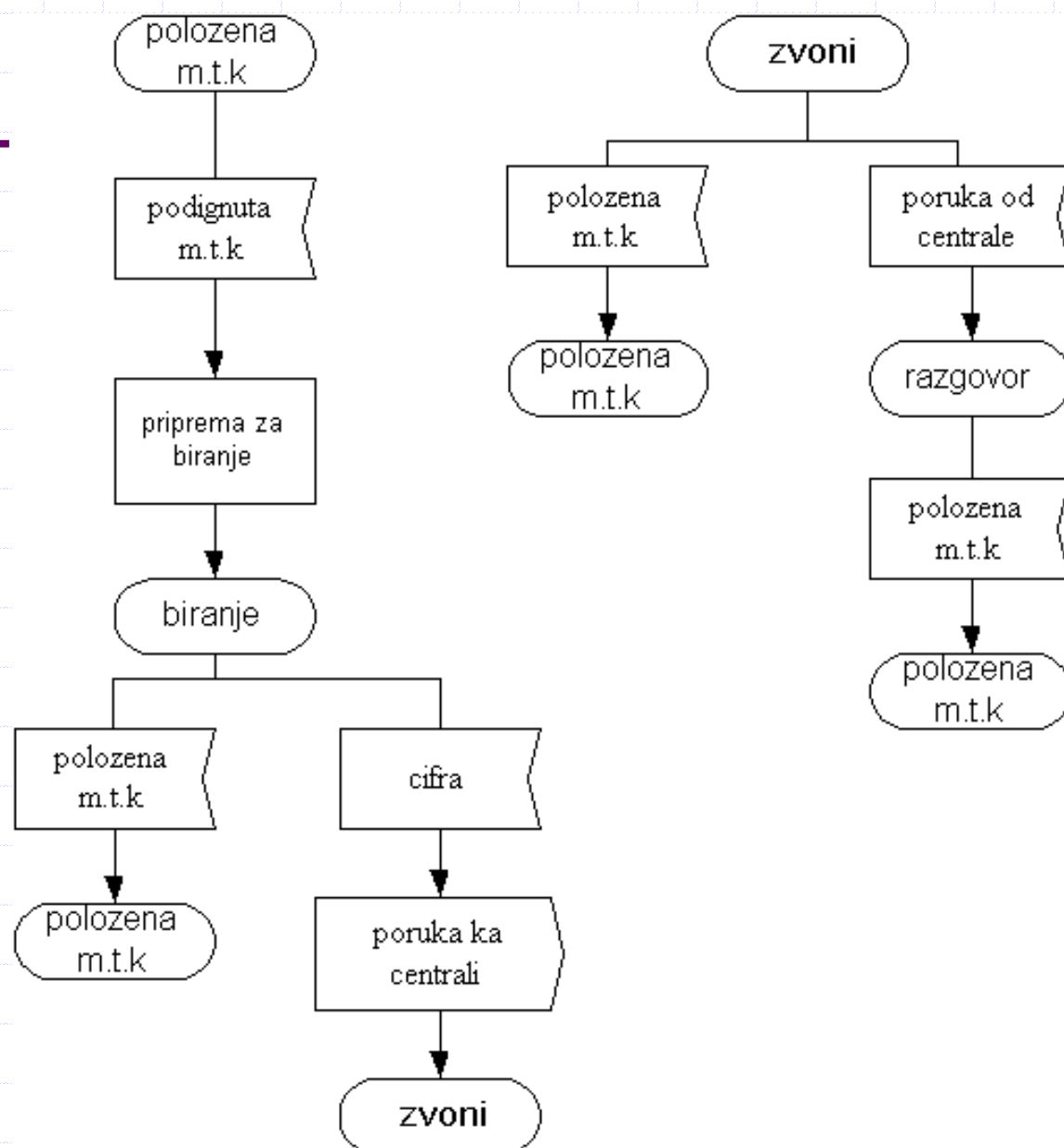
Primer bloka FE1



Primer bloka FE1 - nastavak



SDL za FE1



MSC jezik (*Message Sequence Chart*)

- ◆ Još jedan od načina opisa rada telekomunikacionih sistema je prikaz razmene signala u sistemu.
- ◆ U tu svrhu razvijen je formalni jezik od strane komisije za standardizaciju dat u preporuci Z.120, tkz. MSC jezik.
- ◆ MSC jezik, kao i SDL, ima grafičku i programsku prezentaciju.

MSC jezik - nastavak

- ◆ U razvoju telekomunikacijskih problema od interesa je grafička prezentacija.
- ◆ Naredni primer MSC prikazuje razmenu poruka između funkcionalnih entiteta u slučaju uspostave poziva i raskidanja poziva između dva korisnika.
- ◆ Na ovaj način se bolje mogu sagledati događaji, što korišćenjem SDL jezika nije tako očigledno.

Primer uspostave i raskida poziva

- ◆ Sledi primer uspostave i raskida poziva između dva korisnika.
- ◆ Programska rutina koja nadgleda korisnike, nakon prepoznavanja podignute m.t.k, od strane pozivajućeg korisnika, generiše primitivu SETUP_req koju prosleđuje u funkcionalni entitet FE1.
- ◆ Utvrđenim mehanizmom rada funkcionalnih entiteta SETUP poruka se prosleđuje do funkcionalnog entiteta FE5. Na taj način dolazi se do pozvanog korisnika.

Primer uspostave i raskida poziva

- nastavak

- ◆ Ukoliko je on slobodan ka pozivajućem korisniku se prosleđuje SETUP_resp, te se startuje zvonjava kako pozvanom tako i pozivajućem korisniku.
- ◆ Javljanjem pozvanog prelazi se u stanje razgovora.
- ◆ Nakon obavljenog razgovora pozivajući korisnik polaganjem m.t.k prekida vezu čime se oslobađaju funkcionalni entiteti s njegove strane. Funkcionalni entiteti sa strane pozvanog korisnika se oslobađaju tek kada on položi m.t.k.

Primer MSC-a

