XXVIII JUGOSLOVENSKA KONFERENCIJA ETAN-a, S P L I T, 4 — 8 JUNA 1984. GODINE

Milan ZORIĆ
ELEKTROTEHNIČKI FAKULTET
Zavod za telekomunikacije
41000 ZAGREB, Unska 3

ANALIZA METODA RAZVOJA PROGRAMSKE PODRŠKE KOMUTACIJSKIH SISTEMA MALOG KAPACITETA

ANALYSIS OF SOFTWARE DEVELOPMENT METHODS FOR SMALL SWITHING SYSTEMS

SAŽETAK - Analiziraju se iskustva razvoja programske podrške sistema ETC 960. Diskutira se proces izgradnje programske podrške. Prikazane su specifikacije u jeziku SDL/PR, te su analizirane prednosti takvog pristupa. Programiranje se promatra kao implementacija specifikacija u SDL/PR.

ABSTRACT- Experiance gained while developing system ETC 960 software is analysed. Program production process is discussed. Specifications switen in SDL/PR and their advantages are presented. Programming as implementation of SDL/PR specifications is proposed.

1. UVOD

U procesu razvoja programske podrške komutacijskih sistema važno mjesto zauzimaju formalizirane funkcijske specifikacije. Iskustva sa korištenjem grafičkog oblika jezika SDL za specificiranje funkcija, što dalje predstavlja osnovu za implementaciju u programima, vrlo su pozitivana. Cjelokupan koncept verificiran je tokom razvoja sistema ETC 960.

Pristupajući razradi metoda razvoja programske podrške za novi razvoj mikroprocesorski upravljanog komutacijskog sistema malog kapaciteta, uvode se neke novine u načinu specificiranja funkcija. Jedna od ključnih je i korištenje programskog oblika jezika SDL.

2. NEKA ISKUSTVA U RAZVOJU SISTEMA ETC 960

U procesu razvoja programske podrške sistema ETC 960 bilo je uključeno i istraživanje metoda funkcijske specifikacije i opisivanja, izgradnje modela te postupaka verifikacije njihove korektnosti /1, 2, 3, 8, 9, 10/. Korištena je funkcijska specifikacija na bazi jezika SDL /5, 6/, te su primjenjeni modeli diskretnog automata i Petrijevih mreža. Postojanje formaliziranih specifikacija bit-

no je utjecalo na način pisanja programa, kao i na način njihovog testiranja /1, 4, 11/.

Iskustva stečena u razvoju sistema ETC 960 svakako su pozitivna, a na ovom mjestu ističemo ona najznačajnija.

Inicijalna dekompozicija sistema na više nivoa (upravljanje sklopovima, upravljanje signalizacijom i upravljanje obradom poziva) /1/ osigurava fleksibilnost sistema odnosno novih usluga. S druge strane dekompozicija sistema koja dobro odražava funkcijsku strukturu osigurava lakše definiranje sučelja medju jedinicama sistema te jasnije sagledavanje funkcija odredjenog bloka odnosno pojedinih procesa u bloku.

Korištenje jezika SDL za definiranje formaliziranih specifikacija pokazalo je niz dobrih strana:

- razvoj programa razbija se na dvije faze - definiranje funkcijske specifikacije i preslikavanje specifikacije u program uz odgovarajuću strukturu podataka
- pri izradi funkcijskih specifikacija konstruktor definira funkciju koristeći one mogućnosti koje mu svojim mehanizmom pruža SDL jezik. U tom pogledu postignuta je vrlo pozitivna unifikacija načina razmišljanja i izražavanja u radnom timu
- u prvoj fazi konstruktor nije opterećen detaljima sintakse programskog jezika, mogućih programskih rješenja, strukture i načina kodiranja podataka. Stoga nije čudno da su mnoga rješenja vrlo efikasno i konstruktivno diskutirana u cijelom radnom timu. Ovakve diskusije posebno su doprinijele razvijanju odredjenih saznanja o tome koja su rješenja pogodnija za eventualne buduće izmjene, koja su pogodnija za testiranje itd.
- u fazi realizacije odredjene specifikacije u odgovarajućem programskom jeziku, treba korektno izvršiti preslikavanje definirane funkcijske specifikacije u program. U potpunosti se zadržava struktura definirana funkcijskom specifikacijom, a obzirom da su strukture SDL-a standardne, tada mogu biti usvojeni i standardni programski oblici kojima se pojedini SDL objekt realizira. Posebno je povoljno rješenje u kojem već sistemski programi preuzimaju dio realizacije SDL-a, kao što je slučaj sa procedurnim programima u sistemu ETC 960, gdje sistemski programi detektiraju pojavu ulaza (PRW), te na osnovu stanja veze i ulaza aktiviraju odgovarajući prijelaz (procedurni program).
- testiranje sistema planira se na osnovu funkcijskih specifikacija /3, 4/, te je, obzirom na unificiranu strukturu rješenja moguće unificirati i načine testiranja programa. Time se postiže da je testiranje sustavnije, brže i efikasnije, a nivo testiranosti programa ujednačen za cijeli sistem.

Uz sve dosad navedeno, ipak nije moguće izbjeći pojavljivanje odredjenog broja grešaka. Te se greške mogu klasificirati na mnogo načina /3/, a iskustva iz razvoja sistema ETC 960 pokazuju da je interesantno razlučiti dva tipa grešaka:

- pogrešne funkcijske specifikacije
- pogreška u realizaciji inače ispravne funkcijske specifikacije.

Moguće je spomenuti i greške kod kojih je program ispravan a njegova dokumentacija, uključujući i funkcijsku specifikaciju, nije ispravna. Takve greške najčešće nastaju kada se pri testiranju utvrdi greška pa se programi promijene, a dokumentacija ostane ista, ili kada se naknadno promjene neki zahtjevi, a promjene se provedu samo u programu. U ovom slučaju sistem ispravno radi, ali to može biti izvor novih grešaka prilikom slijedećih modifikacija.

Iskustva pokazuju da je, bez obzira na uzrok nastajanja, te način manifestiranja, greške u realizaciji ispravne specifikacije daleko lakše otkriti. Mnogo je teža situacija sa greškama u funkcijskim specifikacijama. Klasičnim testiranjem vrlo vrlo ih je teško otkriti, jer se obično dešava da način razmišljanja koji je doveo do pogrešne specifikacije, dovede i do takvog plana testiranja koji neće moći ukazati na postojanje greške.

Najbolji načini za smanjivanje broja ovakvih grešaka su:

- precizno definiranje sučelja (SDL opis sučelja, liste signala itd.),
- formalna verifikacija korektnosti barem značajnijih i složenijih dijelova sistema /8, 9, 1o/,
- na ovom nivou tehnologije proizvodnje programske podrške motiviranost i dobra obučenost konstruktora još uvijek presudno utječu na nivo grešaka u programskoj podršci.

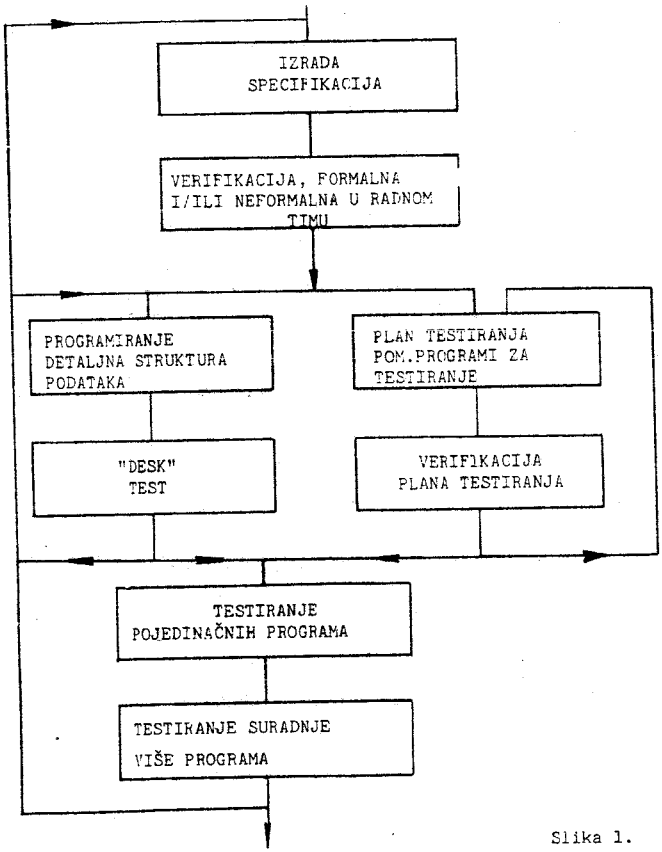
Sve ovo, kao i niz drugih elemenata koji ovdje nisu spomenuti, doprinosi kvaliteti sistema uz povećanje produktivnosti konstruktora i radnog tima u cjelini.

3. PROCES IZGRADNJE PROGRAMSKE PODRŠKE

Po izvršenoj dekompoziciji sistema na blokove i procese unutar blokova /1/, pristupa se razradi pojedinih procesa, pri čemu se slijedi postupak prikazan na slici. Pojedini dijelovi toga procesa opisani su u referencama /2, 3, 4, 8, 9, lo/. Na ovom mjestu želimo istaći važnije momente:

- svi elementi ovog procesa bitno su oslonjeni na specifikaciju u SDL-u,
- svaka aktivnost mora završiti odgovarajućim dokumentima
- plan testiranja radi se paralelno sa izradom programa i to na osnovu specifikacije. Pri tome treba definirati sve podatke potrebne za uspješno testiranje, kao i predvidjeti uvjete testiranja programa, te napisati programe koji se mogu koristiti kod testiranja na razvojnom sistemu,
- kod svih programa pisanih u asembleru tzv. "DESK TEST" ima veliki značaj i morao bi se dosljedno provoditi,
- pored rezultata u obliku programa za dani sistem, važan rezultat je formiranje skupa verificiranih specifikacija koje bi dobrim dijelom trebale biti neovisne o

sistemu, te dogradnja razvojnog sistema za potrebe testiranja.



Slika 1.

4. FUNKCIJSKA SPECIFIKACIJA U JEZIKU SDL/PR

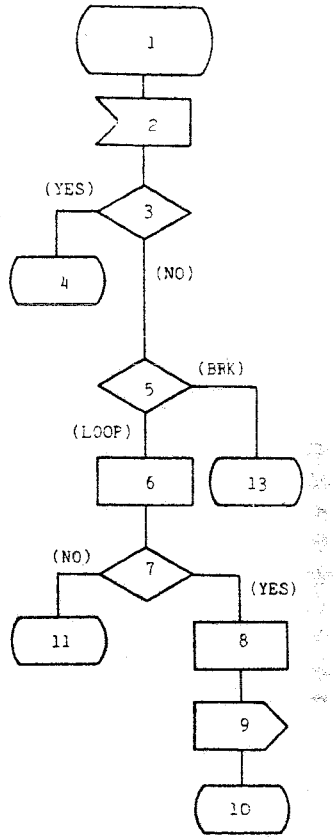
Uz grafički oblik SDL-a, definirani su od strane CCITT-a i slikovni i programski oblik /5, 6, 7/. Sva tri oblika su ravnopravna, a svaki od njih ima odredjene prednosti.

Programski oblik SDL-a definiran je sintaksnim dijagramima i opisom sintakse u Bachus Naur notaciji /7/. Sintaksa SDL/PR može biti proširena tako da se omogući opisivanje slikovnog SDL-a.

Pristupajući definiranju procesa proizvodnje programske podrške za mikroprocesorski upravljani sistem malog kapaciteta, posebno su analizirana svojstva SDL/PR sa ciljem da se on koristi kao radna dokumentacija, uz mogućnost konverzije u grafički oblik u trenutku kada razvoj sistema bude pri kraju. Uočene su sli-

jedeće prednosti:

- brže se piše nego što se crtaju dijagrami
- mnogo je veća gustoća zapisa
- izrazito je veća mogućnost korištenja komentara (inače je uočeno da se u grafičkom obliku komentari slabo koriste, često i zbog prostornih ograničenja)
- lakše je i preciznije preslikavanje programskog oblika u program od grafičkog
- predstavlja osnovu za komentiranje programa na nivou blokova instrukcija (kod realizacije u asembleru)
- predstavlja podlogu za planiranje testiranja programa
- može se lako preslikati u grafički oblik.



Slika 2.

Funkcijska specifikacija u programskom obliku SDL-a može i bez postojanja prevodioca biti pohranjena na računalu što je dobro jer omogućava:

- efikasno manipuliranje sa dokumentima o funkcijskoj specifikaciji,
- specifikacije su svima dostupne što je dobro za sagledavanje sistema u cjelini i interakcije medju pojedinim programima,
- lako je izvršiti promjene u funkcijskoj specifikaciji, čime se povećava vjerojatnost da će korekcije biti provedene kako u programima tako i u specifikacijama.

U ovom trenutku čini nam se da je osnovni problem privikavanja konstruktora na sintaksu SDL/PR (semantika je ista u svim oblicima SDL-a), i to kako pri pisanju specifikacija, tako i pri njihovom čitanju. Pored ovoga, uočeno je da su svi oblici SDL-a slabi za opisivanje složenijih taskova, posebno onih koji moraju imati iterativnu strukturu. Srećom, na nivou obrade poziva takvih taskova ima vrlo malo. U kojoj mjeri taj nedostatak može doći do izražaja kod opisivanja sistemskog dijela i dijela za održavanje i nadzor treba još istražiti. Kako je programski oblik SDL-a (kao uostalom i grafički oblik) još uvijek u fazi razvoja, treba očekivati da će ove slabosti biti u narednom periodu umanjene. Zbog svega

navedenog smatramo da je povoljno da se iskustva u primjeni SDL/PR stiču u relativno malom razvojnom timu i to, kao što je već naznačeno, kao interna radna dokumentacija. U svim komunikacijama izvan radnog tima treba koristiti usvojene načine opisivanja (grafički SDL, dijagrami toka, opisi itd.).
Na ovom mjestu u nastavku se ilustrira način korištenja SDL/PR kroz primjer dijela specifikacije za proces primanja dekadskih impulsa biranja. Na slici 3 dat je programski oblik SDL specifikacije, a na slici 2 prikazan je grafički oblik za stanje WAITDIG.

```
PROCES 'BIRAN';
    STATE '0-SNT' /*[illegible]*/;
        INPUT 'PT' /*[illegible]*/;
            DECISION '[illegible]';
                (YES): NEXTSTATE '0-SNT';
                (NO) : TASK '[illegible]';
                       NEXTSTATE '2-WAITDIG';
            ENDDECISION;
    ENDSTATE '0-SNT';

    STATE '2-WAITDIG' /*WAITING AT DIGIT*/;
        INPUT 'PT';
            DECISION 'EXTA [illegible]';
                (YES): NEXTSTATE '0-SNT';
                (NO) : DECISION 'INFST';
                          (LOOP): OUTPUT 'MARK [illegible]';
                                  TASK 'START [illegible]';
                                  NEXTSTATE '4-STIMP';
                          (BRK) : NEXTSTATE '2-WAITDIG';
                       ENDDECISION;
            ENDDECISION;
    ENDSTATE '2-WAITDIG';

    STATE '4-STIMP' /*START IMPULS*/;
        INPUT 'PT';
            DECISION 'EXTA [illegible]';
                (YES): NEXTSTATE '0-SNT';
                (NO) : DECISION 'INFST';
                          (LOOP): [illegible]
                                  DECISION [illegible]
                                     (YES): TASK [illegible]
                                            [illegible]
                                            [illegible]
                                     [illegible] NEXTSTATE [illegible]
                                  ENDDECISION;
                          (BRK) : NEXTSTATE '2-WAITDIG';
                       ENDDECISION;
            ENDDECISION;
    ENDSTATE '4-STIMP';
```

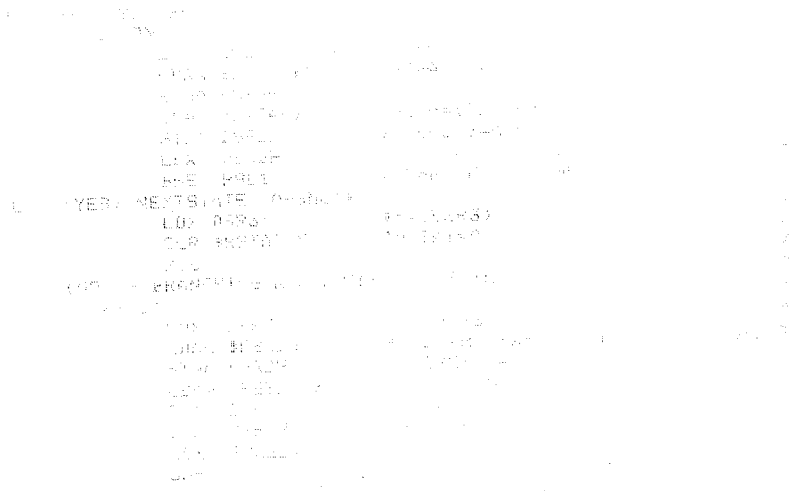
Slika 3.

5. PROGRAMIRANJE KAO IMPLEMENTACIJA SPECIFIKACIJE U SDL/PR

U razvoju mikroprocesorski upravljanog komutacijskog sistema malog kapaciteta planira se korištenje visokog programskog jezika na nivou upravljanja obradom poziva, te assemblerskog jezika na nivou detekcije signala iz okoline odnosno generiranja signala iz centrale.

Struktura visokog programskog jezika je takva da je vrlo mala distanca u odnosu na SDL opis. Stoga na ovom nivou, uz dobro komentiranje programa, nije neophodno korištenje SDL-a.

Kod programiranja u assembleru polazište je specifikacija u SDL/PR. Za preslikavanje specifikacije u SDL/PR u program struktura programa zadržava strukturu specifikacije tako da se elementi SDL/PR specifikacije koriste kao komentari blokova instrukcija u asembleru koje realiziraju dani element specifikacije.



Slika 4.

Pri programiranju se maksimalno trebaju koristiti rješenja standardizirana za sve programe koji sadrža odgovarajući element SDL/PR. Takvim programskim sekvencama moguće je onda posvetiti više vremena u cilju njihovog optimiranja po prostoru i vremenu kao i provjere ispravnosti odredjenog rješenja.
Na slici 4 dan je primjer dijela programa kojim se realizira dio specifikacije dane na slici 3.

6. ZAKLJUČAK

Pristup procesu razvoja programske podrške mikroprocesorski upravljanog komutacijskog sistema malog kapaciteta unaprijedjen je u odnosu na dosadašnju praksu prvenstveno kroz uvodjenje specifikacija u jeziku SDL/PR. Ovim korakom priprema se teren da se, kada bude razvijen prevodilac za SDL/PR, izvrši automatizacija odredjenih faza razvoja /12/, čime bi se bitno povećala produktivnost uz povećanje kvalitete izgradjivanih sistema.

LITERATURA

1. Kunštić,M., B.Mikac, M.Zorić: "Razvoj metoda specifikacije i modeliranja i primjena na sistemu ETC 960", ITA 2 (1983) 1-2, 67-82.
2. Zorić,M.: "Primjena modela diskretnog automata u procesu testiranja programske podrške elektroničkih komutacija", Zbornik radova XXVII ETAN, Struga, 6-lo juna 1983. pp III 249 - III 256.
3. Zorić,M.: "Primjena modela diskretnog automata u procesu testiranja programske podrške elektroničkih komutacija", Magistarski rad, ETF, Zagreb 1982.
4. Marić,V., B.Mikac, K.Ružić, M.Zorić: "Pristup testiranju signalizacijskog podsistema komutacijskog sistema ETC 960", III Simpozij o tehničkoj dijagnostici, Zbornika radova Yurema 28 (1983, 3 svezak, pp 123-126, 1983.
5. "Functional Specification and Description Language (SDL), CCITT Yellow book, Fascicle VI. 7, ITU Geneva 1982.
6. Röckstrom,A., R.Saracco: "SDL-CCITT Specification and Description Language" IEEE Transaction on Communications, Vol. COM-30, No 6, pp 1310-1317, 1982.
7. Draft Recommendation Z. 105 - The program - like form of the SDL (SDL/PR), CCITT, ITU Geneva.
8. Ružić,K., B.Mikac, "Unutarnja signalizacija u procesorski upravljanom komutacijskom čvoru, XXVIII ETAN, Split, 1984.
9. Marić,V., B.Blašković: "Optimalizacija strukture podataka za signalizacijske funkcije, XXVIII ETAN, Split, 1984.
10. Cej,D., I.Lovrek,: "Algoritmi obrade poziva u integriranoj mreži", XXVIII ETAN, Split, 1984.
11. Cej,D., K.Ružić, "Obrada odlaznog i dolaznog poziva u sistemu ETC 960, ITA 2 (1983) 1-2, 131-152.
12. Zorić,M.,:"Automatska programska pomagala u procesu izgradnje telekomunikacijske programske podrške, III Simpozij o tehničkoj dijagnostici, Zbornik radova YUREMA 28 (1983), 3 svezak, pp 135-139, 1983.