Uporaba jezika SDL pri predmetu Programska oprema v TK sistemih

Boštjan Vlaovič, Zmago Brezočnik
Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru
Smetanova ul. 17, SI-2000 Maribor, Slovenija
{*bostjan.vlaovic, brezocnik*}*@uni-mb.si*

Abstract

This paper describes methods used to teach undergraduate students the use of SDL (Specification and Description Language) for design of a simple telecommunication system. The students are encouraged to provide personalized solutions. We are tying to simulate a real-world working environment as much as possible.

1 Uvod

Programska oprema v TK sistemih se poučuje na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru v tretjem letniku visokošolskega strokovnega študijskega programa Elektrotehnika — smer telekomunikacije.

Namen predmeta je seznaniti študente s sodobnimi pristopi pri snovanju, implementaciji ter testiranju programske opreme v telekomunikacijskih sistemih. Cilj vaj je zasnovati sistem skladno z zahtevami, uskladiti nejasnosti z naročnikom ter realizirati sistem s programskim jezikom SDL.

V prvem delu vaj, ki je namenjen spoznavanju delovnega okolja, se dotaknemo tudi osnov Unix okolja in varnega dela na daljavo. Vaje potekajo na operacijskem sistemu Linux, programska oprema pa je nameščena na delovni postaji z operacijskim sistemom Solaris.

V prvi fazi snovanja sistema si pomagamo z opisnim jezikom MSC (Message Sequence Charts). Le-ta nam omogoča jasno in nedvoumno predstavitev množice dogodkov, ki jih pričakujemo v načrtovanem sistemu. Po uspešni uskladitvi zahtev se prične opisovanje učnega primera s SDL-jem.

V prvem delu članka opisujemo programski jezik SDL. Na kratko se dotaknemo zgodovine jezika ter izpostavimo osnovne lastnosti, ki so potrebne za razumevanje vsebine. V nadaljevanju smo se posvetili programskemu orodju ObjectGeode. Le-to študentje uporabljajo pri praktičnem izvajanju vaj. V zadnjem delu članka sledi opis učnega primera in načina dela. Zaključek povzema pridobljene izkušnje in ugotovitve ter oznanja načrte za prihodnje delo.

2 Programski jezik SDL

Specification and **D**escription **L**anguage je standardiziran formalni jezik za specifikacijo in opis sistemov. Razvil in standardiziral ga je CCITT (Consultative Committee of the International Telephone and Telegraph). Danes je zapisan kot standard ITU (International Telecommunication Union) Z.100 [3]. Glavne lastnosti SDL-ja:

- Primeren je za sisteme, ki tečejo v realnem času.
- Programska koda je grafično predstavljena.
- Model je zasnovan na med seboj komunicirajočih procesih. Procesi so opisani s pomočjo razširjenih končnih avtomatov — EFSM (Extended finite state machines).
- Jezik je objektno orientiran.
- Uporaben je vse od zapisa zahtev do končne implementacije.

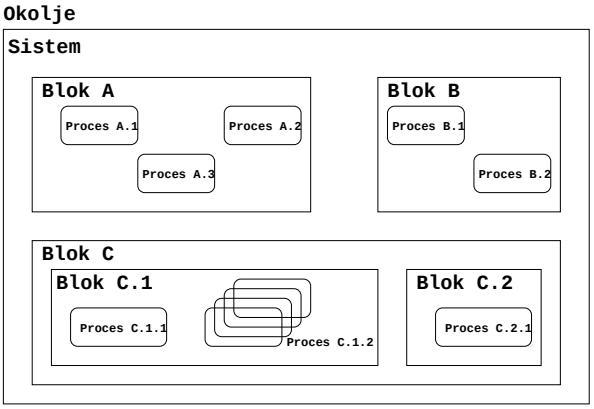
Pri opisu sistema s SDL-jem navadno začnemo z neformalnim opisom na zelo visokem nivoju abstrakcije [2]. Ta specifikacija nudi osnovo za nadaljnji razvoj. Med razvojem se sistem opisuje vedno bolj podrobno. V končni fazi razvoja so formalno opisani vsi deli sistema. Formalno opisane sisteme lahko s posebnimi orodji tudi simuliramo, verificiramo ter avtomatsko generiramo izvršilno kodo za različne ciljne sisteme.

Objekt, ki ga želimo opisati, imenujemo `SISTEM`. Pri opisu sistema s SDL-jem nas prvenstveno zanima njegovo obnašanje. Torej, kaj sistem počne in pri kakšnih pogojih. Vse, kar je zunaj sistema, predstavlja okolje. SDL smatra okolje za črno škatlo, ki spoštuje omejitve sistema. Sistem je lahko zaprt ali odprt. Odprt sistem z okoljem komunicira. Sistemi v telekomunikacijah so odprte narave.

Za opis objektov v SDL-ju obstajata dva sintaktično ekvivalentna zapisa: grafični in tekstovni. Opis objektov lahko razdelimo na tri sklope:

1. arhitektura sistema,
2. opis procesov,
3. določitev podatkovnih tipov.

Najvišji nivo arhitekture predstavlja objekt SISTEM. Sistem je grafično predstavljen kot pravokotnik. Je to, kar SDL specifikacija poskuša definirati. Vse, kar je zunaj sistema (pravokotnika), je okolje (slika 1).



Slika 1: Struktura SDL sistema

Sistem razgradimo na manjše enote, ki jih imenujemo BLOK. Blok predstavlja svojevrsten podsistem. S tem olajšamo opis velikih sistemov. Bloki so med seboj neodvisni. Vsak blok je lahko nadalje razdeljen na podbloke ali pa vsebuje enega ali več PROCESov. Blok tako predstavlja primeren način za združevanje procesov v logične skupine, kot tudi mejo za vidnost podatkov in podatkovnih tipov.

Znotraj bloka, kot tudi z okoljem[1], si procesi izmenjujejo sporočila s pomočjo signalov. Procesi so med seboj povezani s signalnimi potmi. Bloki so med seboj povezani s kanali. Kanal lahko prenaša več signalnih poti. Tudi sistem in okolje sta povezana s kanalom. Teoretično zadostuje že en kanal, praktično pa je kanalov toliko, kolikor je logičnih povezav sistema z okoljem. Kanali in signalne poti so lahko enosmerni ali dvosmerni. Signal je najnižji objekt pri komunikaciji in je definiran z imenom. V primeru, da prenaša podatke, so poleg imena definirani tudi podatkovni tipi prenašanih podatkov. Signal je lahko definiran na nivoju sistema, bloka ali procesa. Signali, ki so definirani na določenem hierarhičnem nivoju, se lahko uporabljajo tudi na vseh nižjih nivojih. Tako se lahko signal, ki je definiran znotraj procesa, uporablja samo med različnimi primerki tega procesa. Signal, ki je definiran na nivoju sistema, pa je uporaben znotraj celotnega sistema, torej povsod. V praksi je pametno upoštevati princip lokalnosti in signal definirati na najnižjem možnem nivoju.

Obnašanje sistema določajo procesi. Procese opisujemo s pomočjo razširjenih končnih avtomatov. Vsak proces je sestavljen iz več stanj. Spremembo stanja povzroči prispeli signal. Vse signale proces pobira iz pridružene vhodne vrste. Vsi signali, ki jih proces sprejme, se postavijo v vhodno vrsto. Signali se obdelujejo po principu FIFO (First In First Out). V novejših verzijah SDL-ja poznamo tudi prioritetne signale. Ob prehodu med stanji se lahko izvedejo različne akcije. Izvrši se lahko računska operacija, starta časovnik, pošlje signal, ...

Med procesi se pogosto prenaša večje število signalov. Zato jih lahko združimo v signalne liste. S tem se predstavitev sistema poenostavi, zmanjšajo pa se tudi potrebni posegi v opise pri nadaljnjem razvoju sistema, saj se ob spremenjeni signalni listi lahko vse popravi na mestu definicije [1].

V primeru, ko proces opisuje obnašanje objekta, ki se pojavlja v več kot enem primerku (npr.: telefonski aparat pri opisu telefonske centrale), opisa ni potrebno ponavljati. Takšnemu procesu se dovoli obstoj v več primerkih. Vsi primerki temeljijo na generični definiciji procesa, vsak primerek pa ima svojo podatkovno strukturo. S tem se množijo tudi signalne poti, ki proces povezujejo z okoljem. Primer takšnega procesa je `Proces C.1.2` na sliki 1.

Procese med seboj ločimo s PID-i. PID (Process Identification Number) nedvoumno določa proces. V primeru procesa `Proces C.1.2`, ko imamo več primerkov enakega procesa, ima vsak primerek svoj PID. Tako lahko pošiljamo signale samo izbranim primerkom procesa.

Vsak proces lahko vsebuje PROCEDURE in STORITVE. Procedure so podobne proceduram v drugih programskih jezikih. Omogočajo združevanje kompleksnih delov v celoto z enkratno definicijo in večkratno uporabo. Procedura je lahko definirana znotraj procesa ali znotraj druge procedure. Definicija procedure lahko vsebuje tudi formalne parametre. Formalni parametri vsebujejo sezname spremenljivk, povezanih s tipom spremenljivk. Procedura bere vhodne signale iz vhodne vrste procesa, v katerem je definirana. Vidna je znotraj objekta, kjer je definirana. Klic procedure se lahko izvrši znotraj prehoda med dvema stanjema.

Storitve omogočajo še natančnejšo razgradnjo procesa. Storitev razvojno orodje ObjectGeode, ki ga uporabljamo pri izvajanju vaj, ne podpira, zato se v našem opisu ne pojavljajo.

Deklaracije spremenljivk in časovnikov se nahajajo v programskih deklaracijah. SDL vsebuje naslednje vnaprej definirane podatkovne tipe: INTEGER, REAL, NATURAL, CHARACTER, ASCII, CHARSTRING, BOOLEAN, PID, TIME, DURATION, ARRAY, STRUCT.

Poleg teh lahko razvijalec definira svoje podatkovne tipe in operacije nad njimi. Za definicijo novih tipov SDL uporablja pristop abstraktnega podatkovnega tipa (ADT — Abstract Data Type).

Vsi procesi v sistemu tečejo vzporedno in neodvisno drug od drugega. Pri klicu procedure se izvajanje procesa ustavi v točki klica. Ko se program vrne iz procedure, se izvajanje nadaljuje v točki za klicem procedure. Procedure znotraj procesa torej ne tečejo vzporedno.

Iz opisanega je razvidno, da lahko isti sistem opišemo na različne načine. Kako sistem razdelimo, je odvisno od različnih kriterijev:

[1] Vse, kar je zunaj mej bloka, je za blok okolje.

- definiranje še obvladljivih blokov glede na velikost in funkcionalnost,
- skladnost z dejansko programsko in strojno opremo,
- naravna funkcionalna delitev,
- ponovna uporabnost,
- minimalno število povezav,
- zdrava pamet.

3 ObjectGeode

Pri praktičnemu delu vaj uporabljamo programsko orodje ObjectGeode proizvajalca Verilog. Podjetje Verilog je v zadnjem mesecu leta 1999 kupil največji tekmec na področju grafičnih programskih orodij za opisovanje telekomunikacijskih sistemov — Telelogic. S tem nakupom je postalo podjetje Telelogic vodilno na trgu programske opreme v telekomunikacijah. Orodje Geode uporablja pri razvoju telefonske centrale SI2000 V5 tudi podjetje Iskratel, d.o.o.

Programski paket ObjectGeode je sestavljen iz naslednjih sklopov:

- OMT Object Editor,
- MSC Editor,
- SDL Editor,
- SDL&MSC Checker,
- SDL&MSC Interactive Simulator,
- SDL&MSC Exhaustive Simulator,
- OMT C++ Code Generator,
- SDL C Code Generator,
- SDL C Run-time Library,
- Design Tracer.

Pri oblikovanju vaj smo se posvetili predvsem MSC in SDL urejevalniku.

4 Učni primer

Ker se študentje pri vajah prvič srečajo s programiranjem v jeziku SDL, smo za učni primer izbrali enostaven telekomunikacijski problem, ki deluje po principu odjemalec-strežnik.

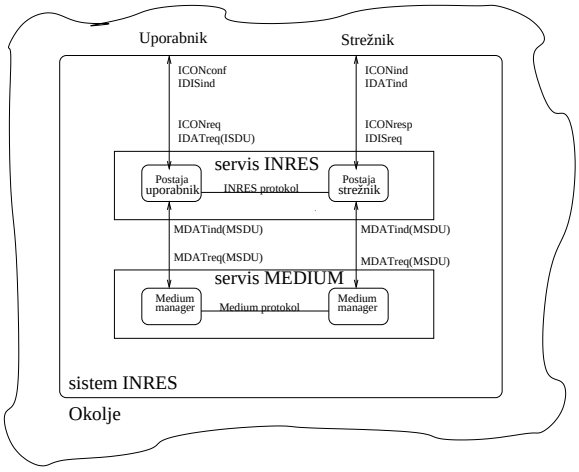
4.1 Zahteve naročnika

Naročnikove zahteve so podane opisno v pripravah na vaje s prostim besedilom. Pri izvajanju vaj predpostavljamo, da naročnik:

- ni seznanjen s postopkom načrtovanja sistema,
- nima tehnične izobrazbe in
- ima jasno vizijo, kaj potrebuje.

Neformalni opis sistema je podan v naslednji obliki: “Potrebujemo sistem, ki bo imel dve priključni točki. Na prvo želimo priključiti uporabnika, na drugo pa centralni bazni sistem. Uporabnik želi na koncu delovnega dne prenesti vse spremenjene in nove podatke v centralni bazni sistem. Iz centralnega baznega sistema do uporabnika se ne prenašajo podatki. Storitev zaradi poenostavitve ni simetrična. Za komunikacijo želimo uporabiti protokol INRES.”

V nadaljevanju je podan opis protokola INRES. Le-ta za prenos uporablja protokol MEDIUM. Obravnavamo torej večslojni pristop opisovanja sistemov (Slika 2).



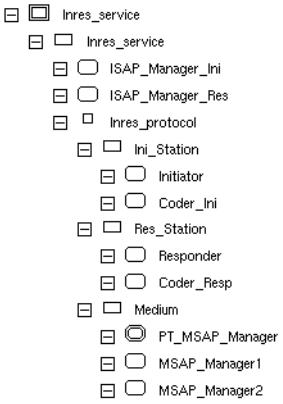
Slika 2: Grafični opis učnega primera

Poznamo horizontalni in vertikalni način razgradnje sistemov. Študentom do določene mere prepustimo izbiro razgradnje sistema. Tako tudi praktično okusijo posledice nepravilnih odločitev, ki so navadno rezultat nenatančnega razmisleka in površne obravnave problema.

4.2 SDL opis sistema

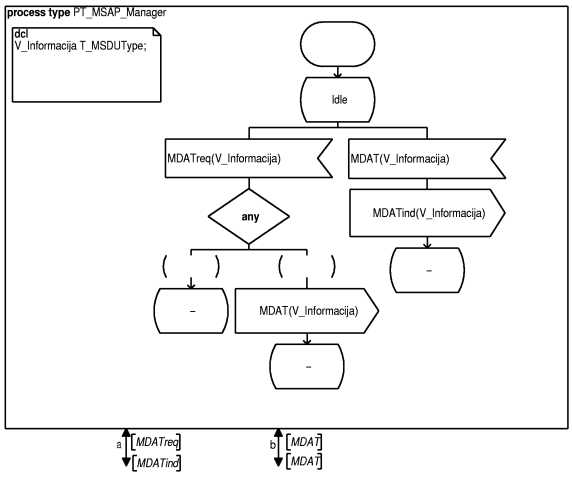
Na vajah želimo opisati celoten sistem. Vključno s servisoma `INRES` in `MEDIUM`. Vsak posameznik se po svoji lastni presoji odloči za končno verzijo razgradnje sistema. Vzpodbujamo originalne rešitve, ki predstavijo posameznikovo sposobnost izvirnega načrtovanja in snovanja sistemov. Izkaže se, da vsi, ki sistem pametno razgradijo, hitreje in enostavneje rešijo problem. Skozi praktično delo ugotovimo, da se gradnja pretirano zapletenih sistemov

ne obrestuje, preprostejši opisi sistema pa povzročijo nemalo težav pri raznih dopolnitvah in spremembah. Kljub temu, da vsi rešujemo isti problem, je ob koncu vaj število rešitev približno enako številu študentov.



Slika 3: Razgradnja učnega primera na bloke in procese

SDL opis sistema, ki je prilagojen prikazu lastnosti SDL-ja, sestavlja pet razširjenih končnih avtomatov. Le-ti so razdeljeni na tri sklope. Najnižji sloj predstavlja servis `MEDIUM`, ki dejansko simulira fizično povezavo med strežnikom in odjemalcem. Zaradi vernejše simulacije realnega okolja predvidimo, da se podatki med prenosom lahko izgubijo. To simuliramo s pomočjo odločitvenega stavka **any** (slika 4). Servis `INRES` zaznava in odpravlja napake ter zagotavlja varni prenos vsem svojim uporabnikom (uporabniku in strežniku). Le-to zagotovimo s ponovno oddajo nepravilno prenesenih "paketov". V primeru ponavljajočih napak predvidevamo, da je prišlo do izpada povezave, zato uporabnika obvestimo o prekinitvi povezave. Kot je razvidno na sliki 2, se za prenos podatkov med slojema uporabljajo različne protokolne podatkovne enote (PDU — Protocol Data Unit).



Slika 4: Opis procesnega tipa za opis servisa `MEDIUM`

Servis `INRES` mora torej skrbeti tudi za pravilno kodiranje in dekodiranje podatkov. Osnovne podatkovne enote so definirane v navodilih za vaje. Množico vmesnih korakov, kot je način in mesto kodiranja, izberejo med načrtovanjem sistema posamezniki — razvijalci.

5 Zaključek

Kljub enostavnosti primera se je izkazalo, da je primeren za poučevanje osnov načrtovanja in razvoja telekomunikacijskih sistemov s programskim jezikom SDL. Študentje se srečajo z množico problemov, za katere je potrebno poiskati primerno rešitev. Med delom vzpodbujamo delo v skupini in komunikacijo, saj se zavedamo, da podjetja od mladega diplomanta poleg osnovnih tehničnih znanj pričakujejo tudi množico osebnostnih lastnosti, kot je sposobnost dela v skupini, jasne predstavitve svojih idej, sposobnost sprejemanja kompromisov in usklajevanje s sodelavci. Med izvajanjem vaj poskušamo ustvariti delovno okolje, ki od študenta pričakuje vse omenjene lastnosti. Pozitiven odnos študentov do takšnega pristopa predstavlja motivacijo za nadaljevanje začrtanih ciljev. Le-ti vsebujejo razširitev učnega primera, natančnejšo definicijo problema ter razdelitev realizacije posameznih funkcijskih sklopov na večje število skupin. Tako bi simulirali tudi sodelovanje med različnimi razvojnimi skupinami znotraj podjetja. V prihodnih letih se želimo podrobneje posvetiti tudi uporabi simulatorja in izdelavi testnih scenarijev s pomočjo opisnega jezika MSC. Tako bi se študentje razdelili na dve osnovni skupini — razvijalce in verifikatorje. Prvi bi skrbeli za razvoj sistema, drugi pa za neodvisno testiranje in preverjanje skladnosti z osnovnimi zahtevami. To bo zahtevalo natančnejše priprave in dodatna usklajevanja. Opisane spremembe bodo povečale zahtevnost vaj, vendar smo prepričani, da bodo študentje na vaje prihajali še z večjim veseljem in motivacijo do dela.

Literatura

[1] Jan Ellsberger, Dieter Hogrefe, and Amardeo Sarma. *SDL Formal Object-oriented Language for Communicating Systems*. Prentice Hall Europe, 1997.

[2] Boštjan Vlaovič. *Generator klicev za telefonsko centralo MLB SI2000 V5: diplomsko delo univerzitetnega študija*. Fakulteta za elektrotehniko, računalništvo in informatiko,, 1999.

[3] ITU-T Recommendation Z.100. CCITT specification and description language (SDL), 1993. Series Z: Programming Languages.