Dragutin Kermek

Fakultet organizacije i informatike
Varaždin

UDK: 681.3.06
Stručni rad

DDE - novi izazov modularnosti dizajna i programiranja

Razvoj i uvođenje novih radnih okolina donosi promjene u dizajnu i programiranju. Te promjene uglavnom su inovatorskih osobina pa se prema tome prvi puta susreću u navedenim fazama životnog ciklusa softvera. Radna okolina s najvećim utjecajem na području osobnih računala, Microsoft Windows 3.0, uvela je novu tehniku prijenosa podataka između aplikacija pod nazivom dinamička izmjena podataka (Dynamic Data Exchange, DDE). Primjenom te tehnike moguće je proširiti pojam modularnosti s dosadašnjeg pojma modula kao integralnog dijela cjeline (nivo objektne verzije koda) na modul u vidu zasebne aplikacije (nivo izvršne verzije koda) koji putem DDE prima i/ili šalje potrebne podatke. Moduli povezani pomoću DDE mogu se poistovjetiti s klasama u objektno orijentiranoj paradigmi. Sličnost proizlazi iz ideje klase kao zasebne cjeline pri čemu korisnici ne moraju poznavati njenu strukturu.

Dinamička izmjena podataka (DDE); komunikacija; radna okolina; modul; objektna orijentacija; Smalltalk/V; KnowledgePro.

1. RAZLOZI ZA MODULARNOST

U dvije faze životnog ciklusa softvera, dizajnu i programiranju, javljaju se tri pojma koji mogu imati blisko značenje. Njihova sličnost može dovesti do određenih problema ukoliko nisu jasno razgraničeni. To su:

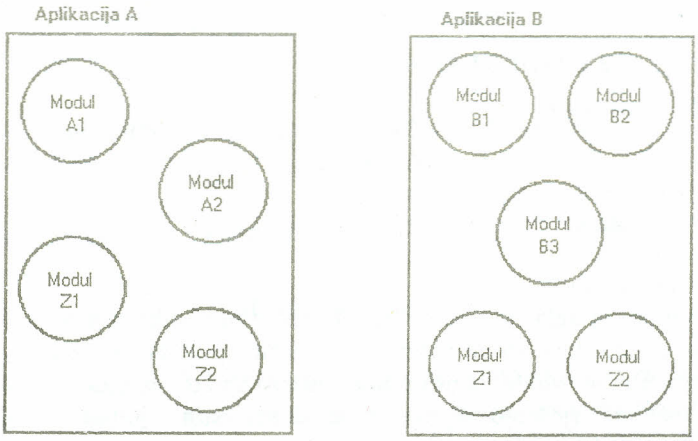
- program
- modul
- aplikacija.

Pod programom uglavnom se podrazumijeva skup kodiranih istrukcija za računalo. Te instrukcije pohranjene su na nekom od nosilaca podataka. Uz program nije nužno vezana funkcionalnost. Modul je nezavisni i zaokruženi dio cjeline koji mora imati svoju funkciju u konačnom rješenju. Aplikacija je funkcionalna cjelina sastavljena od barem jednog modula. Međusobni odnosi tih pojmova mogu biti različiti. Modul može biti spremljen u više programa, odnosno, u jednom programu može se nalaziti više modula. Odnos modul-program može donekle ovisiti o složenosti modula, ali odluka o tom

odnosu većinom je prepuštena programeru. Odnos modul-aplikacija ovisi o složenosti problema koji se komjputerizira, s time da bitno utječe i kreativna sposobnost dizajnera za prepoznavanje dijelova koji se mogu izdvojiti iz cjeline i proglasiti modulima. Odnos program- aplikacija može se promatrati kroz odnose program-modul i modul- aplikacija. Konačno, može se reći da je program programerska kategorija, a modul i aplikacija su dizajnerske kategorije.

Primjena modularnosti (izgradnja kompleksne cjeline iz manjih i jednostavnijih dijelova s ograničenim funkcijama,tj. modula) poboljšava efekte rada (brzina, kvaliteta i sl.). Put prema uvođenju modularnosti započeo je, iz praktičnih razloga, s pozicije programiranja. Dva su razloga za to: diletantska izrada aplikacije s početkom na programiranju ili izrada manjih aplikacija kod kojih se svjesno pokušava izbjeći dizajn. Tek kasnije uočava se da je za kvalitetno rješenje nužan povratak iz faze programiranja na dizajn. Primjenom modularnosti u dizajnu dobije se kvalitetna podloga za programiranje.

Svrha primjene modularnosti bit će prikazana kroz izradu dviju aplikacija (A i B) u vremenskom slijedu. Početni zadatak koji treba izraditi projektni tim predstavlja kompjuterizaciju jedne jednostavne funkcije. Rješenje, aplikacija, sastoji se od jednog programa. Slijedeći zadatak ima funkciju sličnu prethodnom zadatku. Npr. utvrđeno je da je sličnost njihovih podfunkcija (npr. kontrola štampača, ekrana i sl.). To znači da je moguće obaviti čisto preuzimanje koda iz programa prethodnog rješenja, s tim da se novo rješenje nadopunjuje za razliku nove funkcije. Situacija je slijedeća: raspolaže se s dvije aplikacije pri čemu svaka u cijelosti zadržava svoj kod iako postoji presjek njihovih podfunkcija. Na slici 1 prikazana je gornja situcija.



Slika 1. Smještaj modula po aplikacijama

S ovakvim rješenjem jasno je da bi bilo potrebno provesti izmjene u kodu obje aplikacije čak i kod najmanje promjene u zajedničkim podfunkcijama. Ponavljanje istih izmjena u kodu na većem broju aplikacija idealno je za generiranje pogrešaka, a jedan od razloga je monotonost takvog posla.

Izlaz je pronađen tako da se u fazi dizajna obavlja dekompozicija zadatka na module, pri čemu svaki modul obavlja manji dio cjelokupne funkcije aplikacije. U ovom slučaju gotovo dolazi do jednakosti između modula i programa. Dekompozicija na module nije rutinski posao ukoliko se želi postići puni efekt primjene modularnosti.

Iskustvo je pokazalo potrebnim uvođenje određenih kriterija vezanih uz dizajn sustava. Neki od njih tretiraju problematiku modula. To su:

- povezanost modula (coupling) - mjera nezavisnosti između modula u kompjuterskom programu (ANSI/IEEE STD 729/1983, preuzeto iz Strahonja90)
- kohezija modula (cohesion) - stupanj funkcionalne povezanosti elemenata unutar jednog modula (Strahonja 90).

Situacija unutar modula nije od važnog interesa za prikaz DDE pa zato neće biti obrađivan kriterij kohezije modula. Za dizajn se smatra da je uspješnije obavljen ukoliko je manja povezanost između modula. Razlozi su slijedeći (Strahonja 90):

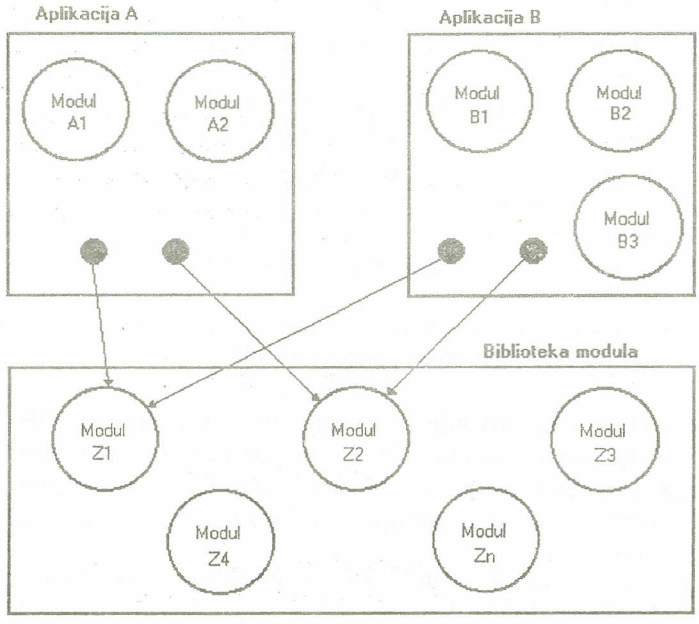
- manje su šanse za utjecaj greške u jednom modulu na drugi modul (RIPPLE efekt)
- manji je rizik da promjena u jednom modulu zahtijeva promjene i u drugom modulu (HYDRA efekt)
- manja je potreba poznavanja detalja drugih modula kod održavanja
- smanjuje se kompleksnost održavanja, a povećava jednostavnost i razumljivost.

Realizacija može biti različita. Primitivnije rješenje sastoji se od ugrađivanja različitih INCLUDE ili COPY komandi u programe, što ovisi o sintaksi ciljnog programskog jezika, čime se modul uključuje u rješenje. Ovim rješenjem izbjegnuto je višestruko mijenjanje u pojedinim aplikacijama. Ostala je potreba prevođenja svih programa koji uključuju taj modul, kao i završno povezivanje svake aplikacije.

Elegantnije rješenje je izgradnja biblioteke objektnih modula. Modul se prevodi samo jednom i uključuje se u biblioteku. Ostaje povezivanje svake aplikacije s novom verzijom modula. Slika 2 prikazuje korištenje biblioteke objektnih modula.

Modul se u oba slučaja tretira kao dio cjeline, aplikacije, i ne može djelovati zasebno. Razlog tome je problem prijenosa ulaznih i izlaznih podataka modula. Izuzetak je modul koji je potpuno nezavisan, tj. nema ulaznih ni izlaznih veza prema drugim modulima. Takav modul može biti proglašen aplikacijom (zasebno prevođen i povezan)

ukoliko ciljni programski jezik aplikacije dopušta pozivanje (zamjenjivanje) nove aplikacije iz trenutno aktivne aplikacije (SWAPPING).



Slika 2. Smještaj modula uz korištenje biblioteke modula

Za modul koji zahtijeva ulazne i/ili daje izlazne podatke nije bilo moguće primijeniti navedenu tehniku u postojećim programskim jezicima i radnim okolinama.

Nove radne okoline primijenjene na prednostima postojeće opreme (PC AT 286, 386 ili 486, Macintosh) otvaraju nove poglede na prikazani problem. Rješenje je predočeno u novoj tehnici koja omogućava prijenos podataka između aplikacija, a naziva se dinamička izmjena podataka (**Dynamic Data Exchange, DDE**), a radi pod radnom okolinom Microsoft Windows. Primjenom DDE otvara se mogućnost izdvajanja pojedinih modula u zasebne aplikacije, čime se postiže puni efekt modularnosti. Dvije aplikacije mogu komunicirati putem DDE ako su obje izrađene za rad u Windows okolini i ako su dizajnirane za podršku DDE. Prva pretpostavka nije dovoljna, mada ona omogućava povezivanje aplikacija i njihovu komunikaciju putem DDE bez posebne kontrole od strane korisnika.

Na tržištu postoji već nekoliko proizvoda koji podržavaju DDE. To su:

- Word - Microsoft
- Excel - Microsoft
- Smalltalk/V Windows - Digitalk
- KnowledgePro Windows - Knowledge Garden.

Korisnici ovih proizvoda mogu koristiti prednosti DDE i međusobno ih povezati. Na taj način svaki od njih može postati aplikacija- modul nekom drugom proizvodu iz te skupine.

2. SLIČNOST DDE I KOMUNIKACIJA

U slučaju DDE više odgovara termin konverzacija između aplikacija nego prijenos podataka između njih. To je jasnije kada se kaže da se prijenos podataka putem DDE temelji na principima komunikacija. Da bi konverzacija bila moguća, u najopćenitijem slučaju, moraju postojati barem dvije strane, u terminologiji komunikacija to su klijent (client, koji inicira konverzaciju, tj. traži podatke) i server (server, daje podatke). Zahtijeva li situacija, moguća je komunikacija većeg broja klijenata s jednim serverom, odnosno, jedan klijent može komunicirati s većim brojem servera. Konačno, moguća je situacija kada je klijent istovremeno i server (ne sam sebi).

Ljudsko komuniciranje između klijenta i servera obavlja se po određenim pravilima (procedurama). Na početku slijedi upoznavanje (pozdravljanje). Uspije li to, moguć je daljnji nastavak, tj. konverzacija. Po obavljenom razgovoru slijedi pozdrav za rastanak i prekid razgovora. To isto vrijedi i za komunikacije između aplikacija. Aplikacija-klijent šalje poruku za iniciranje komunikacije. Primi li ona potvrdan odgovor, znači da je moguća konverzacija, tj. otvoren je kanal za konverzaciju. Prestanak potrebe za konverzacijom označava se slanjem poruke za zatvaranje (terminiranje) komunikacijskog kanala. Promatra li se način DDE komunikacija, može se reći da postoji velika sličnost između aplikacije servera i modula.

3. NAČIN KOMUNICIRANJA IZMEĐU APLIKACIJA PUTEM DDE

Komunikacija među aplikacijama započinje njihovom identifikacijom, tj. otvaranjem komunikacijskog kanala. Taj postupak obavlja klijent, a za servera je važno da je aktivan. Identifikacija je potrebna zbog mogućnosti prisustva većeg broja komunikacijskih sudionika, tj. klijenata i servera. Ključ za identifikaciju mora biti jednoznačan, a sastoji se od:

- naziva aplikacije (**Application name**)
- naziva subjekta (**Topic name**).

Za klijenta uspješno uspostavljanje komunikacije sa serverom rezultira vraćanjem odgovarajućeg DDE identifikacijskog broja (**DDE handle**). Svaki otvoreni DDE kanal ima svoj jednoznačan DDE broj. Sve ostale komunikacijske aktivnosti koje obavlja klijent s jednim serverom, preko njihovog DDE kanala ,odvijaju se uz navođenje DDE broja tog kanala.

Za neposrednu konverzaciju između aplikacija uvodi se treći dio, pod imenom "naziv stavke" (**Item name**). Tim dijelom određuje se sama konverzacija. Klijentu stoji na raspolaganju nekoliko mogućnosti:

- može zahtijevati podatke od servera
- može zahtijevati informacije o promjeni određenog podatka kod servera, ali bez slanja nove vrijednosti. Takva veza naziva se **Warm Link.**
- može zahtijevati opskrbu novom vrijednosti kada se određeni podatak promijeni. Takva veza naziva se **Hot Link.**
- može zahtijevati od servera izvršavanje određene komande.

Prekid komunikacije između klijenta i servera može se obaviti na dva načina:

- klijent zatvara DDE kanal
- server prekida s radom.

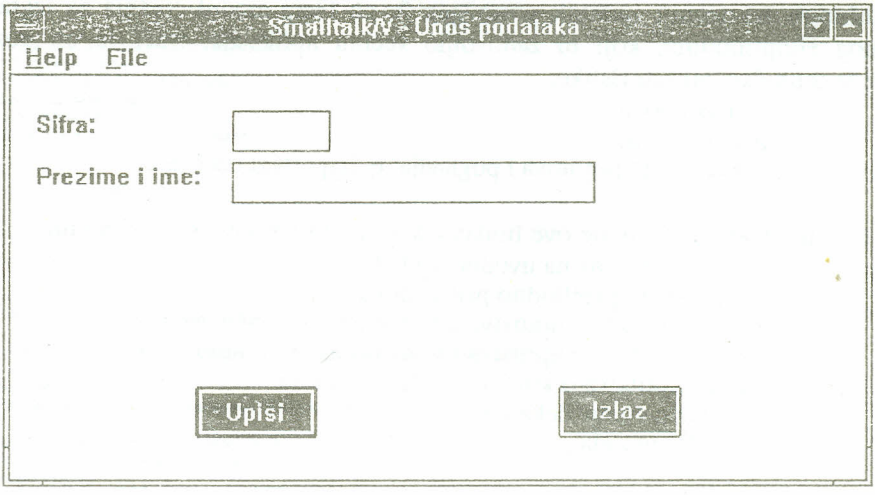
Drugi slučaj predstavlja nepoželjan način prekida, posebno ako se promatra sa stajališta klasičnog modula, koji ne poznaje takav oblik prekida.

4. PRIMJER KOMUNIKACIJA PUTEM DDE

Za potrebe daljeg prikaza DDE uvodi se demo aplikacija za jednostavni unos podataka. Osim unosa podataka, u aplikaciju je potrebno uključiti i interaktivnu pomoć u radu (Help-modul). Analizirajući problem Help-modula u fazi dizajna, uočava se da on predstavlja idealan primjer za izgradnju općeupotrebljivog modula. Prilog tome je i sličnost Help-modula velikog broja aplikacija. Njegovim izdvajanjem iz navedene aplikacije nastaju dvije zasebne cjeline, a veza između njih bit će uspostavljena putem DDE.

4.1 Zahtjevi postavljeni dizajnom i izbor programskog jezika

Navedeno je da će to biti jednostavan unos podataka pa se zbog toga unose samo šifra, prezime i ime. Aplikacija za unos podataka ima svrhu prikaza načina aktiviranja aplikacije Help- modula (prema standardu, pritiskom tipke F1). Za postizanje preglednosti neće biti uključene funkcije kao što su spremanje, pretraživanje i sl, koje nemaju direktnu vezu za prikaz DDE. Ekranski izgled prikazan je na slici 3.



Slika 3. Izgled prozora za unos podataka

Uspoređivanjem različitih programskih jezika koji podržavaju DDE (C++, Turbo Pascal, Smalltalk, KnowledgePro i dr.) izdvojen je Smalltalk kao najpodesniji za ispunjavanje postavljenog zadatka za unos podataka. Osim posjedovanja navedenih uvjeta potrebnih za DDE, Smalltalk je čisti objektno orijentirani programski jezik. Objektno orijentirana osobina olakšava dublji ulaz u modularnost jer se svaka klasa promatra kao modul. Potpunost, tj. čistoća objektne orijentacije dodatna je snaga modularnosti zbog nemogućnosti da se komunikacija između objekata ostvari na drugi način osim slanjem poruka. Dalje, Smalltalk pruža svojim korisnicima pregled u 99% svog programskog koda, putem svoje razvojne okoline.

Klasičan, proceduralni pojam modula poistovjećuje ga sa složenim funkcijama ili procedurama (tj. potprogramima). Gledajući tako, moglo bi se zaključiti da je modul kod objektno orijentiranih programskih jezika (OOPJ) jednak metodi, što nije istina. Modul je kod OOPJ podignut na viši nivo. On je vezan uz vrstu podataka, a preko njega i na njegove metode. Drugim riječima, moguće je postaviti jednakost:

modul = tip.

B.Mayer (Mayer 88) smatra da je "fuzija tih dviju naoko različitih notacija ono što daje objektno baziranom dizajnu značajnu prednost." Tokom implementacije u OOPJ navedena prednost dolazi do posebnog izražaja. Program 1 sadrži programski kod aplikacije za unos podataka.

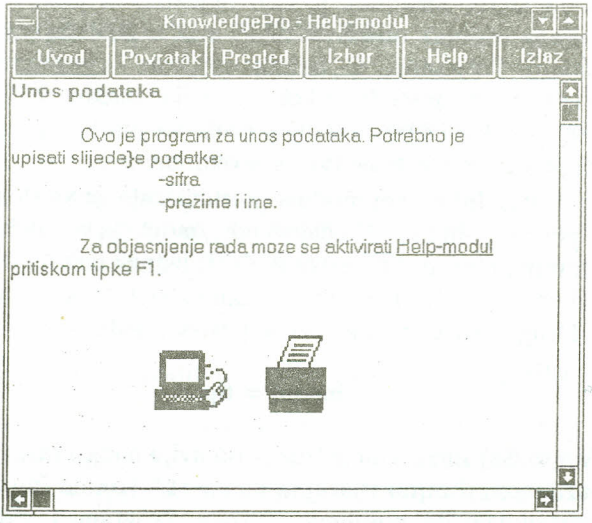
Mnogo zahtjevniji zadatak postavljen je na Help-modul. Težnja za izgradnjom idealnog Help-modula, koji bi zadovoljio većinu aplikacija, traži da on posjeduje slijedeće tehničke karakteristike:

- prikaz teksta
- prikaz grafike
- povezivanje pojmova i poglavlja, tj. hypertext osobine.

Dizajnom su postavljene ove funkcije koje mora izvršavati Help-modul:

- direktni prijelaz na uvodno uputstvo
- povratak na prethodno prikazano uputstvo
- izbor određenog uputstva upisivanjem njegovog naziva
- izbor određenog uputstva prema popisu svih uputstava
- prikaz uputstva o korištenju Help-modula
- prijelaz u stanje čekanja
- prekidanje rada.

Ekranski izgled Help-modula prikazan je na slici 4. Programski jezik u kojem će biti izrađena aplikacija za unos podataka, Smalltalk, ne posjeduje hypertext osobine (što ne znači da ih nije moguće dograditi). Prema tome, potrebno je izabrati drugi programski jezik za Help-modul. Izabran je KnowledgePro jer posjeduje sve tražene osobine, a uz to, on je i OOPJ. Program 2 sadrži programski kod Help-modula. Kostur dijela programa za hypertext preuzet je iz originalnog programa za Help iz paketa KnowledgePro.



Slika 4. Izgled prozora za Help-modul

PROGRAM 1

```
1
2 ViewManager subclass: #DdeTestAplikacija
3   instanceVariableNames:
4     'ddeServer ddeClient sifra naziv '
5   classVariableNames: ''
6   poolDictionaries:
7     'WinConstants VirtualKeyConstants ' !
8
9
10 !DdeTestAplikacija class methods ! !
11
12
13
14 !DdeTestAplikacija methods !
15
16 aktivirajHelp
17     "Aktiviranje Help modula tipkom F1"
18
19     ddeServer notNil ifTrue:  [ ddeServer updateExportedItem: 'testakt' object: '1'] !
20
21 close: aPane
22     "Zatvaranje aplikacije i terminiranje DDE kanala"
23
24     ddeServer notNil ifTrue: [ ddeServer terminate] .
25     ddeClient notNil ifTrue:   [ ddeClient terminate] .
26     super close.!
27
28 initChannel
29         "Iniciranje kanala za testiranje DDE komunikacije"
30
31     ddeServer := DDEServer newServer: self
32         application:'DDETestAplikacija' topic:'Unos podataka'.
33     ddeServer updateExportedItem: 'testakt' object: '0'.
34     ddeServer updateExportedItem: 'testvar' object: 'Unos podataka'.
35     ddeClient := DDEClient newClient: self
36         application: 'Kpwin' topic: 'f:\ kpwin\ helpapl.kb'.
37     ddeClient notNil ifTrue:
38       [ ddeClient pokeItem: 'ddeapp' object: 'DDETestAplikacija'.
39         ddeClient pokeItem: 'ddetopic' object: 'Unos podataka'.
40         ddeClient pokeItem: 'txtfile' object: 'f:\ kpwin\ testapl.hlp'.
41         ddeClient pokeItem: 'ndxfile' object: 'f:\ kpwin\ testapl.ndx'.
42         ddeClient pokeItem: 'ddeitem' object: 'testvar'.
43         ddeClient pokeItem: 'ddestart' object: 'testakt'.
44         ddeClient pokeItem: 'ddeuvod' object: 'Uvod'.
45         ddeClient pokeItem: 'ddeizbor' object: 'Izbor'.
46         ddeClient pokeItem: 'ddehelp' object: 'Help'.
47         ddeClient executeCommand: '!!main:aktiv_dde().'] !
48
49 initWindowSize
50         "Privatno - Postavljenje velicine prozora"
51     ^(500 @280)!
52
53 openTest
54     "Otvaranje aplikacije"
55
56     | lineHeight charWidth charHeight |
57
58     lineHeight := TextFont height + 4.
59     charHeight := TextFont height.
60     charWidth := TextFont width.
61
62     self labelWithoutPrefix: 'Smalltalk/V - Unos podataka'.
63     self menuWindow addMenu: self sectionMenu owner: self.
64     self mainView when:#close perform:#close:.
65     self initChannel.
66
67     self addSubpane:
68         {StaticText new leftJustified;
69             contents: 'Sifra:';
```

```
70          framingBlock: [ :box |
71              Rectangle new leftTop: (charWidth * 2) @(lineHeight * 1)
72                  extent: (charWidth * 15) @(charHeight*1) ] ).
73      self addSubpane:
74          (sifra := FormattedEntryField new setTextLimit: 6;
75              framingBlock: [ :box |
76                  Rectangle new leftTop: (charWidth * 16) @(lineHeight * 1)
77                      extent: (charWidth * 8) @(charHeight*1.5) ] ).
78      self addSubpane:
79          (StaticText new leftJustified;
80              contents: 'Prezime i ime:';
81              framingBlock: [ :box |
82                  Rectangle new leftTop: (charWidth * 2) @(lineHeight * 2.5)
83                      extent: (charWidth * 15) @(charHeight*1) ] ).
84      self addSubpane:
85          (naziv := FormattedEntryField new setTextLimit: 30;
86              framingBlock: [ :box |
87                  Rectangle new leftTop: (charWidth * 18) @(lineHeight * 2.5)
88                      extent: (charWidth * 30) @(charHeight*1.5) ] ).
89      self addSubpane:
90          (Button new contents: '&Upisi';
91              when: #clicked perform: #upisiPodatak: ;
92              defaultPushButton;
93              framingBlock: [ :box |
94                  Rectangle new leftTop: (charWidth * 15) @(lineHeight * 9)
95                      extent: (charWidth * 10) @(lineHeight * 1.5)] ).
96      self addSubpane:
97          (Button new contents: '&Izlaz';
98              when: #clicked perform: #close:;
99              framingBlock: [ :box |
100                 Rectangle new leftTop: (charWidth * 45) @(lineHeight * 9)
101                     extent: (charWidth * 10) @(lineHeight * 1.5)] ).
102
103     self openWindow!
104
105 sectionMenu
106     "Postavljanje menu-a za aplikaciju"
107
108     ^Menu new
109         appendItem: 'Help F1' selector: #aktivirajHelp accelKey: F1Key accelBits: nil ;
110             title: '&Help'.!
111
112 upisiPodatak: aPane
113
114     sifra contents: ''.
115     naziv contents: ''! !
```

PROGRAM 2

```
dde ().
!helpdir is 'f:\ kpwin\ '.
:ddestatus is [ ] .
:txtfile is ''.  :ndxfile is ''. :ddeapp is ''.:ddetopic is ''.
:ddeItem is ''.:ddestart is ''.:ddeuvod is ''.:ddeizbor is ''.
:ddehelp is ''.:helpitem is ''.:helpstart is ''.
:mainFont is create_char_font ( [ 1,1,400,'F','F','F',0,1,34,'Helv'] ).
:hyperFont is create_char_font ( [ 1,1,400,'F','T','F',0,1,34,'Helv'] ).
:boldFont is create_char_font ( [ 1,1,700,'F','F','F',0,1,34,'Helv'] ).
:bigFont is create_char_font ( [ 1.5,1.2857,400,'F','F','F',0,1,34,'Helv'] ).
if string_copy (?!helpdir, string_length (?!helpdir), 1)  '\ '
   then !helpdir is concat (?!helpdir,'\ ').
:wMain is window (Finish,15,2,60,23,'KnowledgePro - Help-modul',
   [ overlapped, siblings, showChildren,thickFrame,TitleBar, MinimizeBox,
   ControlMenu, MaximizeBox] , , , , [ close_event, resize_event]  ).
:hand is load_mouse_cursor ( concat ( ?!helpDir, 'hand.cur')).
hyper_cursor (?hand).
:helpIcon is  load_icon ( concat ( ?!helpdir, 'kpwin.ico' ) ).
attach_icon (?wmain,?helpIcon).
:wB is window ( , 1,1, element(window_info (?wMain), 3),2, ,
   [ child, visible, siblings, showChildren] , ?wmain, , gray).
:poz is element ( window_info (?wMain), 3) / 6.
:b1 is button (Uvod,uvod,1,1,?poz).
:b2 is button (Povratak,povratak,?poz + 1,1,?poz).
:b3 is button (Pregled,pregled,(?poz * 2) + 1,1,?poz).
:b4 is button (Izbor,izbor,(?poz * 3) + 1,1,?poz).
:b5 is button (Help,help,(?poz * 4) + 1,1,?poz).
:b6 is button (Izlaz,izlaz,(?poz * 5) + 1,1,?poz).
:w1 is window ( , 1,3, element ( window_info (?wMain), 10),
   element ( window_Info (?wMain), 11)-2,,[ child, siblings, horzScroll,
   vertScroll,showChildren, visible]  ,?wmain).
if last (system_info ())  2
   then :color is green2
   else :color is black.
hyper_display (?color,,?hyperFont).
:list is [ ] .
:menuSet is yes.
show_window ( ?wMain, 7).
wait ().

topic Uvod.
   set_focus (?w1).
   if last (?list) is Uvod then exit ().
   list gets Uvod.
   mark (?ddeuvod).
   set_active_window (?w1).
end. (*Uvod *)

topic Povratak.
   set_focus (?w1).
   if list_length (?list) = 1
      then Uvod ()
   else
      list is sublist (?list, 1, list_length (?list) - 1) and
      item is last (?list) and
      list is sublist (?list, 1, list_length (?list) - 1) and
      (if ?item is Uvod
         then Uvod ()
         else mark (?item)).
end. (* Povratak *)

topic Pregled.
   :temp is window (,30.57,8.562,45,6.562,,
[ popup,showChildren,siblings,DialogFrame] ).
   make_modal (?temp).
   text ('#y2  Pregled uputstva:').
   ed1 is edit_line (,ok,2.25,3.8,25,char_event).
   button (Ok,ok,32,1.5,9).
   button (Cancel,cancel,32,4.25,9).
```

```
70  show_window ( ?temp ).
71  set_focus (?ed1).
72
73  topic ok (i,e,h).
74   set_focus (?w1).
75   if one_of ( [ 13,ok] , ?i)
76     then info is get_text (?ed1) and
77          set_display_window (?w1) and
78          mark ( ?info ) and
79          close_window (?temp) and
80          set_active_window (?w1) and
81          ok is t.
82  end.
83
84  topic cancel.
85   set_focus (?w1).
86   close_window (?temp).
87  end.
88 end. (* Pregled *)
89
90 topic Izbor.
91  set_focus (?w1).
92  if last (?list) is ?ddeizbor then exit ().
93  mark (?ddeizbor).
94 end.
95
96 topic Help.
97  set_focus (?w1).
98  if last (?list) is help then exit ().
99  mark (?ddehelp).
100 end.
101
102 topic Izlaz.
103  menuSet is yes.
104  show_window ( ?wMain,2).
105 end.
106
107 topic Finish (info,event) .
108  do (?event).
109
110  topic close_event.
111    if not(?ddestatus is [ ] )then
112       dde_close(?ddestatus) and
113       close ([ ?ndxfile,?txtfile] ).
114   close_window (?w1).
115   continue ().
116  end.
117
118  topic resize_event (width, height, poz, ind).
119    if ?menuSet is yes then menuSet is no and exit ().
120    menuSet is yes.
121    hide_window ( [ ?w1,?wb] ).
122    :width is element ( window_info (?wMain), 3).
123    :height is element ( window_info (?wMain), 4).
124    if ?height  15 then :height is 15.
125    if ?width  60 then :width is 60.
126    :poz is ?width / 6.
127    resize_window (?wMain, ?width, ?height).
128    resize_window (?wb, ?width, 2).
129    :ind is 1.
130    while ?ind <=  6 then
131      move_window (?concat('b', ?ind), (?poz * (?ind - 1)) + 1, 1) and
132      resize_window (?concat('b', ?ind), ?poz, 2) and
133      ind is ?ind + 1.
134    move_window (?w1, 1, 3).
135    resize_window (?w1, element ( window_info (?wMain),10),
136 element ( window_info ( ?wMain), 11) - 2).
137    show_window ([ ?wb,?w1] ).
138    menuSet is no.
139  end.
140 end. (* Finish *)
```

```
141
142 (* ============================ MARK ========================== *)
143
144 topic mark (item).
145   if ?txtfile is '' then exit ().
146   set_file_pos (?ndxfile,0).
147   :indexInfo is read (?ndxfile,concat ('//',?item),'//').
148   if list_length (?indexInfo)  2 or ?indexInfo is number_to_char (26)
149     then hypertextError ()
150     else set_file_pos (?txtfile, first (?indexInfo)) and
151          :t is read_char (?txtfile, element (?indexInfo, 2)) and
152
153          set_text (?w1, [ #e#rb,?item,#rd, ?t]  ) and
154          list gets (?item).
155
156   topic hypertextError.
157     :temp is window (,21.57,8,46,9,,[ popup,showChildren,siblings,DialogFrame] ).
158     make_modal (?temp).
159     use_font (?boldFont).
160     text ('
161
162   There is no further information on this item.').
163     b1 is button (Continue, continue,18,6,11).
164     show_window (?temp).
165     set_focus (?b1).
166     wait ().
167     close_window (?temp).
168     use_font (?mainFont).
169     set_focus (?w1).
170   end. (* hypertextError *)
171 end.  (* Mark *)
172
173 topic b.
174   use_font (?boldFont).
175 end.
176
177 topic d.
178   use_font (?mainFont).
179 end.
180
181 topic l.
182   use_font (?bigFont).
183 end.
184
185 (* ============================ DDE ========================== *)
186
187 topic aktiv_dde.
188   set_file_pos ([ ?txtFile,?ndxFile] ,0).
189   ddestatus is dde_open (test, ?ddeapp, ?ddetopic).
190   if ?ddestatus is [ ] then  exit ().
191   dde_advise (?ddestatus, ?ddestart).
192   dde_advise (?ddestatus, ?ddeitem).
193
194   topic test (info, event, handle).
195   do (?event).
196     topic dde_data_event.
197       if element(?info, 2) = ?ddestart then helpstart is element(?info, 1).
198       if element(?info, 2) = ?ddeitem then helpitem is element(?info, 1).
199       if ?helpstart is '1' and not(?helpitem is '') then
200         set_active_window (?wMain) and
201         show_window (?wMain,1) and
202         !main:mark (?helpitem) and
203         helpstart is '0'.
204     end.
205
206     topic dde_ok_event.
207     end.
208     topic dde_fail_event.
209     end.
210   end. (* test *)
```

4.2 Objašnjenje rada aplikacije za unos i Help-modula

Aplikacija za unos podataka pridružena je klasi **DdeTestAplikacija**. U hijerarhiji Smalltalk klasa ova klasa predstavlja podklasu klase ViewManager.

Za uspješno povezivanje aplikacije za unos podataka i Help-modula potrebno je prvo aktivirati program Help-modula (u ovom primjeru on se naziva helpapl.kb, a smješten je na direktoriju f:kpwin). Njegovim aktiviranjem definiraju se početni elementi, kao što su:

- otvaranje mogućnosti za DDE
- nekoliko vrsta fontova
- veličina i raspored prozora u kojem će Help-modul biti aktivan
- hypertrxt osobine:
- hypertext značka (cursor)
- hypertext font
- hypertext boje i dr.

Inicijalizacija završava prebacivanjem prozora u pozadinski (hide) status i prijelazom u stanje čekanja (P2L1- linija 1 u programu 2). Sada je Help-modul spreman za kasnije korištenje. Slijedi aktiviranje bazne aplikacije izvršavanjem Smalltalk izraza:

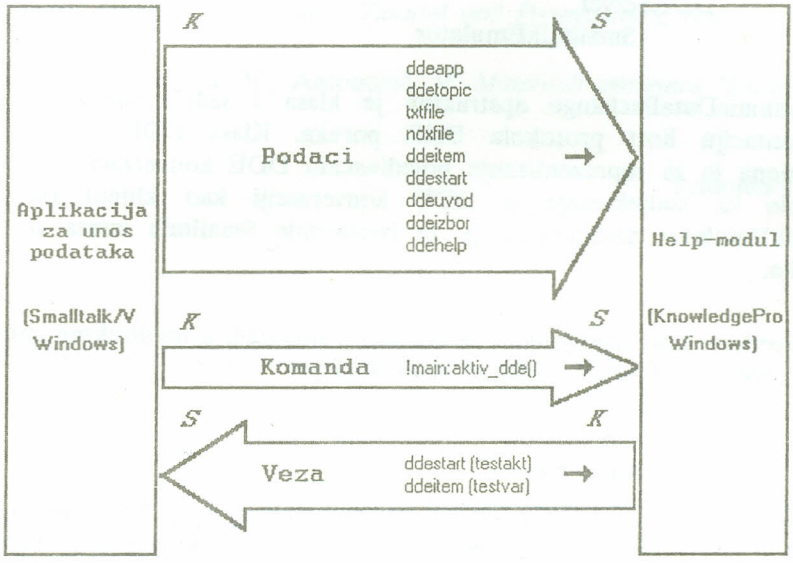
DdeTestAplikacija new openTest.

- Metodom openTest obavljaju se slijedeće radnje:
- određuje se naziv prozora za aplikaciju (P1L62)
- definira se menu aktivnost (P1L63)
- definira se način završavanja rada (P1L64)
- šalje se poruka initChannel (P1L65)
- postavlja se izgled prozora za aplikaciju (P1L66-101)
- aktivira se aplikacija porukom openWindow (P1L103).

Potrebno je posvetiti pažnju poruci initChannel (P1L88). Metoda initChannel vlastita je metoda klase DdeTestAplikacija. Preko nje ostvaruje se inicijalizacija kanala za DDE kojim će se prenositi vrijednosti varijabli testakt i testvar (značenje tih varijabli bit će kasnije objašnjeno) (P1L31-34).

Slijedi otvaranje kanala prema Help-modulu (serveru) kojim će se prenijeti vrijednosti u parametrizirane varijable: ddeapp, ddetopic, txtfile,... (P1L35-46). Metoda završava slanjem komande !main:aktiv_dde() Help-modulu, što znači da će se izvršiti topic tog naziva. U tom topic-u otvaraju se datoteke podataka i ključeva prema vrijednostima parametriziranih varijabli txtfile i ndxfile (P2L88). Slijedi otvaranje kanala za preuzimanje vrijednosti varijabli kojima se vrši aktiviranje Help-modula i određivanje naziva uputstva koje se traži (P2L184). U Smalltalk-u to su ranije navedene varijable

testakt i testvar. Uspješno otvoren kanal znak je da je moguće postaviti veze (linkove) za te varijable (P2L191-192). Topic test kontrolira sve DDE događaje kada je Help-modul klijent. Poziv topic-a dde_data_event označava da je obavljen prijenos podataka. Ukoliko varijabla, čiji je naziv spremljen u helpstart (testakt), ima vrijednost "1" i ako je vrijednost varijable, čiji je naziv spremljen u helpitem (testvar), različita od "", obavlja se aktiviranje prozora Help-modula i prikaz odgovarajućeg uputstva pozivom topic-a !main:mark(?helpitem) (P2L199-202). Slika 5. prikazuje način komuniciranja putem DDE. Ideja za sliku preuzeta je iz slike 6. iz LaLonde91.



Slika 5. Grafički prikaz veza između klijenta i servera

Help-modul aktivira se iz aplikacije za unos podataka pritiskom tipke F1. Izlaz iz Help-modula znači njegov povratak u stanje čekanja, a kontrola se vraća aplikaciji za unos. Programski dio postupka komunikacija putem DDE mogao je biti jednostavniji. Mogle bi se izbaciti varijable ddeitem, ddestart, ddeuvod, ddeizbor i ddehelp. Njihova je namjena postizanje što veće fleksibilnosti, tako da programer nije obavezan primjenjivati strogo određene varijable za kontrolu aktiviranja DDE i strogo određene vrijednosti za obavezna poglavlja u uputstvima (Uvod, Izbor, Help).

4.3 Podrška za DDE u jezicima Smalltalk i KnowledgePro

Otvorenost Smalltalk-a pruža idealne uvjete za razumijevanje komunikacija putem DDE. Pregledom odgovarajućih klasa i njihovih metoda može se vrlo detaljno upoznati način komunikacije putem DDE. Korisniku stoje na raspolaganju slijedeće klase:

- DynamicDataExchange
  - DDEClient
  - DDEServer
    - SmalltalkEmulator.

DynamicDataExchange apstraktna je klasa i sadrži specifične metode za implementaciju host protokola DDE poruka. Klasa DDEClient (DDEServer) namijenjena je za reprezentiranje pojedinačnih DDE konverzacija. Ona osigurava protokole za sudjelovanje u DDE konverzaciji kao klijent (server). Klasa SmalltalkEmulator namijenjena je za izvršavanje Smalltalk izraza iz neSmalltalk aplikacija.

Navedene klase namijenjene su korisniku Smalltalk-a, no direktnu implementaciju DDE-a obavljaju slijedeće podklase klase Window:

- DDEAuxWindow
  - DDEAuxWindow
  - DDEAuxServer.

Korisnik nema vidljive potrebe za instancama ovih klasa. DDEAuxWindow koristi se preko klase DynamicDataExchange za pristup host sustavu. DDEAuxClient (DDEAuxServer) koristi se preko klasa DDEClient (DDEServer) za olakšanje korištenja DDE-a. One su uvedene u namjeri rasterećenja korisnika od poznavanja postupka prijenosa putem DDE koji ovisi o okolini implementacije. Potpuno poznavanje DDE podrazumijeva detaljno upoznavanje klasa prve i druge grupe. KnowledgePro nema tako pregledan način korištenja DDE. Kod njega taj se postupak odvija pozivanjem odgovarajućih topic-a.

ZAKLJUČAK

Primjenom DDE otvaraju se nove mogućnosti za modularnost. Modul postaje aplikacija, a veza između modula-aplikacija prebačena je na DDE. Komunikacija putem DDE specifična je zbog toga što je moguć dvosmjeran prijenos podataka među aktivnim aplikacijama. Bitno je da se komunikacija obavlja bez posebne kontrole korisnika. Ta kontrola ostavljena je kontrolnom modulu radne okoline.

LITERATURA:

1. **LaLonde91**: LaLonde, W., Pugh, J. *Dynamic data exchange in Smalltalk/V Windows,* Journal of Object-oriented programming, vol.4, no. 1, 1991, str: 56-61
2. **KnowledgePro90**: *KnowledgePro (Windows) Tutorial and Programming Handbook*, KnowledgePro Garden, 1990.
3. **Meyer88**: Meyer, B. *Object-oriented Software Construction*, Prentice Hall, 1988.
4. **Smalltalk91**: *Smalltalk/V Windows Tutorial and Programming Handbook*, Digitalk, 1991.
5. **Strahonja90**: Strahonja, V., Antonović, V. *Materijali seminara "Dizajn IS"*, FOI Varaždin 1990.

Primljeno: 1991-07-15

Kermek D. DDE - A new Challenge for Design and Programming Modularity

SUMMARY

The paper reviews a new technique for passing information between applications: Dynamic Data Exchange, DDE. The facility is designed to permit an integrated community of applications to work together on a common problem. It works on one environment: Windows Microsoft. Applying the idea of DDE in the design stage, a module becomes an application. This increases the power of modularity in a new way - the modularity of applications. To present the DDE facility, the paper provides one example with two applications. The main aplication (server) is in Smalltalk/V and Help-module (client) is in KnowledgePro.