UDC: 004.41 INFO M: str. 17-24

SPECIFIKACIJA STRUKUTURA POSLOVNIH APLIKACIJA U ALATU IIS*CASE
A SPECIFICATION OF THE STRUCTURES OF BUSINESS APPLICATIONS IN THE IIS*CASE TOOL

Aleksandar Popović, Ivan Luković, Sonja Ristić

REZIME: Cilj ovog rada jeste da prezentira nove koncepte i alate, ugrađene u alat IIS*Case, koji su namjenjeni definisanju struktura poslovnih aplikacija. Svrha tih unapređenja alata IIS*Case jeste stvaranje osnova za izgradnju generatora kôda transakcionih programa i aplikacija za rad nad bazama podataka. U radu je prezentiran novi koncept pod nazivom poslovna aplikacija, kao i alat Business Application Designer putem koga projektant može kreirati specifikacije struktura poslovnih aplikacija. Opisane su izražajne mogućnosti novih koncepata, kao i osnovne funkcionalnosti alata Business Application Designer.
KLJUČNE REČI: Softversko inženjerstvo, tip forme, poslovna aplikacija, CASE alat.
ABSTRACT: A goal of this paper is to present new concepts and tools, embedded into the IIS*Case tool, that are aimed to define the structures of business applications. The purpose of such improvements of the IIS*Case tool is to make foundations for the development of a code generator for transaction programs and applications that are executed over a database. In this paper we present a new concept named business application, as well as the Business Application Designer tool by means of a designer can create the specifications of the business application structures. We discuss expressiveness of new concepts and present basic functions of Business Application Designer.
KEY WORDS: Software engineering, form type, business application, CASE tool.

1. UVO

Cilj ovog rada je da prezentira dio rezultata rada na višegodišnjem projektu razvoja projektantskog CASE alata, pod nazivom *Integrated Information Systems CASE Tool* (IIS*Case). Na sadašnjem stepenu razvoja, podržane su projektantske aktivnosti vezane za:

- konceptualno projektovanje šeme baze podataka, transakcionih programa i poslovnih aplikacija informacionog sistema,
- automatizovano projektovanje podšema relacione baze podataka u trećoj normalnoj formi,
- automatizovanu integraciju podšema u jedinstvenu šemu baze podataka u trećoj normalnoj formi i
- automatsko generisanje SQL opisa šeme baze podataka [1, 8, 9, 10, 13].

Cilj istraživanja opisanih u ovom radu jeste unapređenje alata IIS*Case koje stvara osnove za izgradnju generatora kôda transakcionih programa za rad nad bazama podataka. Prethodne verzije alata IIS*Case projektantu nijesu pružale mogućnost formalnog opisa funkcionalnosti vezanih za uzajamne odnose generisanih ekranskih formi. Model koji ne sadrži specifikaciju tog dijela informacionog sistema ne predstavlja adekvatnu osnovu za izgradnju generatora kôda. Jedan od ciljeva istraživanja opisanih u ovom radu jeste obogaćivanje IIS*Case-a novim konceptima putem kojih će to biti moguće. Kao rezultat istraživanja uveden je novi koncept pod nazivom *poslovna aplikacija*. U ovom radu opisana je opšta struktura poslovnih aplikacija u alatu IIS*Case, kao i načini specifikacije struktura konkretnih poslovnih aplikacija, koje su u skladu sa tom opštom strukturom. Pored toga, opisan je alat koji je ugrađen u IIS*Case i čija je namjena da podrži navedene projektantske aktivnosti. Ovaj programski modul omogućava projektantu da na integrisan i vizuelno orijentisan način specificira strukturu poslovnih aplikacija, tj. omogućava mu da definiše koje će sve forme učestvovati u toj poslovnoj aplikaciji, kao i da definiše kako se forme međusobno povezuju.

Rad, pored uvoda i zaključka, ima četiri odjeljka. U drugom odjeljku izvršena je uporedna analiza izražajnih mogućnosti koncepata IIS*Case-a, s jedne strane, i koncepata na kojima počivaju alati *DeKlarit* i *Oracle Designer*, s druge strane. Pored toga, dat je kratak prikaz MDA metodološkog pristupa i apostrofirane sličnosti sa metodološkim pristupom na kojem se zasniva IIS*Case. U trećem odjeljku ukratko je opisan koncept tipa forme na kojem se zasniva modelovanje putem alata IIS*Case. U četvrtom odjeljku prezentiran je koncept poslovne aplikacije, kao i neki prateći koncepti koji su bitni za definisanje struktura poslovnih aplikacija. U petom odjeljku opisan je alat *Business Application Designer*. Prikazane su osnovne funkcionalnosti ovog alata i opisana je njegova uloga u kreiranju specifikacija koje se odnose na strukturu poslovnih aplikacija jednog informacionog sistema.

2. PREGLED STANJA U RELEVANTNOJ OBLASTI

Cilj izlaganja datog u ovom odjeljku jeste uporedna analiza srodnih alata i metodologija u relevantnoj oblasti. Istaknute su glavne razlike i sličnosti pristupa opisanog u ovom radu i pristupa koje koriste alati *Oracle Designer* i *Deklarit*, u pogledu definisanja struktura poslovnih aplikacija. Jednim dijelom, odjeljak je posvećen MDA metodologiji i rezultati istraživanja prikazani su u kontekstu MDA metodološkog pristupa.

Oracle Designer je integrisani CASE alat namjenjen, između ostalog, projektovanju šema baza podataka i modelovanju poslovnih aplikacija nad tim bazama podataka [6]. Ovaj alat podržava projektantske aktivnosti vezane za sve faze životnog ciklusa softvera, kao što su modelovanje pro-

cesa sistema, modelovanje šeme baze podataka, automatsko generisanje kôda, itd. Takođe, ovaj alat pruža mogućnost kompletnog dizajniranja poslovnih aplikacija. Koristeći *Oracle Designer* projektant kreira module koji predstavljaju finalnu specifikaciju budućih transakcionih programa (sa opisom ekranske ili izvještajne forme) ili menija. Koncept modula u određenoj mjeri korespondira sa konceptom tipa forme na kojem se zasniva projektovanje putem alata IIS*Case. Definisanju modula, svakako, mora da prethodi definisanje šeme baze podataka [6]. Za razliku od pristupa koji podržava *Oracle Designer*, projektant koristeći IIS*Case simultano specificira i opis ekranskih formi i strukturu šeme baze podataka, tj. projektant kreirajući tipove formi specificira ekranske ili izvještajne forme transakcionih programa, a posredno, kreira i polazni skup atributa i ograničenja. Zadate specifikacije tipova formi transformišu se u opis šeme baze podataka, a takođe predstavljaju osnovu za generisanje transakcionih programa za rad sa bazom podataka.

Jedan modul kreiran putem alata *Oracle Designer* može se posmatrati i kao segment poslovne aplikacije namjenjene jednom tipu radnog mjesta. Ovaj alat pruža mogućnosti da se specificira kako se kreirani moduli objedinjavaju u poslovnu aplikaciju, ali ne postoji zaseban koncept putem kojeg je moguće formalno imenovati i opisati neku poslovnu aplikaciju informacionog sistema. Pojedinačni dijelovi strukture poslovnih aplikacija definišu se odvojeno, na nivou modula i ne postoji zaseban alat koji je namjenjen da podrži projektantske aktivnosti specificiranja strukture aplikacije, već se u te svrhe koriste funkcije alata *Design Editor*. Putem ovog alata različiti segmenti poslovne aplikacije definišu se zasebno, korišćenjem različitih koncepata. Za razliku od ovog pristupa, IIS*Case posjeduje posebno izdvojenu, namjensku funkciju za definisanje struktura poslovnih aplikacija. Polazni skup koncepata proširen je novim konceptima koji su namjenjeni isključivo definisanju struktura poslovnih aplikacija i realizovan je zaseban alat koji podržava aktivnosti kreiranja tih specifikacija.

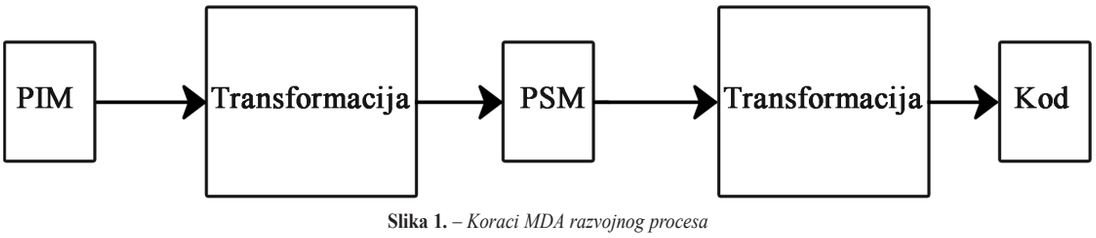
Alat *DeKlarit* namjenjen je agilnom razvoju softverskih komponenata informacionih sistema. Poslovna komponenta je glavni koncept na kojem se zasniva modelovanje putem ovog alata i on u velikoj mjeri korespondira sa konceptom tipa forme na kojem se zasniva modelovanje putem IIS*Case-a. Poslovna komponenta je nenormalizovana hijerarhijska struktura koja se sastoji od atributa i ključeva i reprezentuje izgled buduće šeme baze podataka [2]. U skladu sa pretpostavkom o postojanju šeme univerzalne relacije, svaki atribut jedinstveno je identifikovan svojim imenom. Ovaj alat je u stanju da generiše SQL kôd za različite sisteme za upravljanje bazama podataka. Takođe, na osnovu specifikacija poslovne komponente, moguće je generisati transakcioni program. Ovaj CASE proizvod nema poseban koncept, kao ni poseban alat koji se bavi strukturiranjem generisanih transakcionih programa u jednu jedinstvenu poslovnu aplikaciju. Koncept tipa forme u IIS*Case-u je semantički bogatiji, jer postoji izdvojen koncept koji je namjenjen strukturiranju poslovnih aplikacija, kao i zaseban alat čija je namjena definisanje tih struktura.

Posljednjih godina, važno mjesto u ovoj oblasti zauzimaju pristupi razvoju softverskih sistema, pa i informacionih sistema, koji u svom fokusu imaju modele sistema [5, 11]. Model više nije samo sredstvo za analizu, projektovanje i dokumentovanje sistema, već u sebi sadrži kompletnu semantiku sistema, što je osobina i programskog kôda. Postoji više pristupa koji se temelje na ovoj ideji, a najpoznatiji je *Model Driven Architecture* (MDA), koji je standardizovan od strane *Object Mangement Group* (OMG) organizacije.

MDA pristup podrazumijeva podizanje nivoa apstrakcije prilikom modelovanja i razdvaja *Platform-Independent* modele (PIM) koji su nezavisni od implementacione platforme i *Platform-Specific* modele (PSM) koji u sebi nose informacije o implementaciji [5, 11]. Na slici 1. prikazana su tri glavna koraka MDA razvojnog procesa.

U idealnom slučaju PIM model transformiše se u jedan ili više PSM modela, koji se potom transformišu u programski kôd u konkretnom programskom okruženju. Danas, postoje alati koji pružaju mogućnost transformisanja navedenih modela u programski kôd u konkretnom programskom okruženju. Ipak, u velikom broju slučajeva, oni generišu samo dijelove programskog kôda**.** [11].

Rational Rose je jedan od takvih alata. On je vizuelno orijentisan i namjenjen je modelovanju i analizi objektno-orijentisanih sistema putem UML familije jezika. Pored toga, ovaj alat nudi i mogućnost generisanja programskog kôda. Kako je UML jezik opšte namjene, koristeći njegove izražajne mogućnosti, moguće je modelovati široku klasu sistema, ali je teško opisati neke konkretne karakteristike sistema, kao što su, na primjer, strukture poslovnih aplikacija, ili vizuleni atributi ekranskih formi nekog informacionog sistema. Zato je programski kôd, generisan od strane alata *Rational Rose*, neophodno dopuniti dijelovima koji se odnose na organizaciju i funkcionalnost ekranskih formi, što zahtijeva dodatno angažovanje programera. IIS*Case je alat specijalno namjenjen modelovanju informacionih sistema, tj. modelovanju šema baza podataka i poslovnih aplikacija za rad nad tim bazama podataka. To znači da koncepti na kojima se zasniva IIS*Case nisu opšte namjene, već su domenski specifični. Na osnovu modela informacionog sistema iskazanog putem takvih koncepata, moguće je generisati kompletan programski kôd poslovnih aplikacija, koji neće zahtijevati naknadne, ozbiljnije intervencije programera.



Slika 1. – *Koraci MDA razvojnog procesa*

Postoji sličnost MDA pristupa i pristupa projektovanju putem alata IIS*Case [8, 10 11, 13]. Modelovanje korišćenjem alata IIS*Case vrši se na visokom nivou apstrakcije, jer projektant, koristeći koncepte na kojima se IIS*Case zasniva, opisuje informacioni sistem ne specificirajući implementacione detalje. Tako dobijeni model informacionog sistema može se svrstati u klasu PIM modela. Različiti učesnici projekta ili na bilo koji način zainteresovani subjekti, imaju potrebu da sistem sagledaju sa različitih aspekata (tačaka gledišta). Tako na primjer, naručilac, korisnik, projektant ili programer imaju različite poglede na dati informacioni sistem. Pogled (*view*) [11, 15], u ovom kontekstu, predstavlja jednu reprezentaciju informacionog sistema. Svaki pogled opisuje se putem „jezika" koji je razumljiv onome kome je pogled namjenjen. Ovaj „jezik" naziva se aspekt (tačka gledišta, *viewpoint*) [11, 15]. Njega, između ostalog, čine koncepti i pravila za njihovo strukturiranje, putem kojih se može opisati pogled, apstrahujući sve detalje sistema nebitne sa stanovišta datog učesnika ili zainteresovanog subjekta. Koncepti IIS*Case-a dovoljno su izražajno bogati i moguće ih je grupisati u različite, ne nužno disjunktne podskupove koncepata. Ovi podskupovi koncepata i pravila za njihovo strukturiranje mogu se tretirati kao različite tačke gledišta na sistem (*viewpoints*), što omogućava sagledavanje informacionog sistema sa različitih aspekata. Na taj način, moguće je generisati različite poglede (*views*) na dati informacioni sistem. Svaki od pogleda reprezentuje neku od struktura informacionog sistema, na način koji je primjeren učesniku ili zainteresovanom subjektu projekta kojem je namjenjen. Neki od pogleda koje je moguće generisati primjenom IIS*Case-a su:

– izgled i funkcionalnost ekranskih formi koje čine korisnički interfejs,
– dijagram poslovnih aplikacija,
– šema relacione baze podataka,
– graf zatvaranja šeme baze podataka i dr.

Neki od pogleda mogu se svrstati u klasu PIM modela, dok drugi pripadaju klasi PSM modela. Dosadašnji i budući razvoj alata IIS*Case usmjeren je ka omogućavanju transformacija platformski nezavisnih u što veći broj platformski zavisnih pogleda. Na sadašnjem stepenu razvoja IIS*Case-a, podržano je automatsko generisanje relacione šeme baze podataka, njena transformacija u implementacioni opis šeme baze podataka namjenjen izvršavanju na konkretnom sistemu za upravljanje bazama podataka, a u toku je i izrada generatora kôda čija je namjena transformisanje zadatih specifikacija u programski kôd transakcionih programa. Dalji pravci razvoja odnose se i na uvođenje novih i proširivanje postojećih koncepata što će omogućiti formiranje novih gledišta na sistem, a samim tim i novih pogleda na druge aspekte informacionog sistema, kao što su poslovni procesi ili hardversko-softverska infrastruktura. Takođe, dalja istraživanja usmjerena su na transformaciju dobijenog modela u druge PSM modele, kao što su XML šeme za opis šema baza podataka, objektno-relacione šeme baza podataka, itd.

3. KONCEPT TIPA FORME

Projektant putem alata IIS*Case kreira model informacionog sistema. Sve projektantske specifikacije vezane za jedan informacioni sistem čuvaju se u okviru jednog projekta. Aplikativni sistem predstavlja osnovni činilac nekog projekta i on može da sadrži proizvoljan broj tipova formi. Tip forme je centralni koncept na kojem počiva modelovanje putem IIS*Case-a. On predstavlja generalizaciju poslovnih dokumenata, a samim tim i ekranskih ili izvještajnih formi koje korisnik upotrebljava u svrhu komunikacije s informacionim sistemom. Za razliku od tradicionalnog pristupa projektovanju, u kojem projektovanje šeme baze podataka prethodi izradi ekranskih formi transakcionih programa, putem IIS*Case-a projektant prvo specificira ekranske forme transakcionih programa i indirektno, kreira inicijalni skup atributa i ograničenja. Skup tipova formi predstavlja jedan platformski nezavisan pogled na informacioni sistem iz perspektive kranjeg korisnika. IIS*Case podržava automatsko generisanje šeme relacione baze podataka i njenog grafa zatvaranja, na osnovu specificiranih tipova formi. Na taj način projektant specificiranjem tipova formi istovremeno projektuje i šemu buduće baze podataka i izgled korisničkog interfejsa. Nadalje, IIS*Case omogućava generisanje implementacionog opisa šeme baze podataka, kao i programskog kôda ekranskih formi transakcionih programa za rad nad tom bazom podataka. Tako se dobijaju dva nova pogleda na informacioni sistem, koji su, za razliku od ranije pomenutih, platformski zavisni.

Jedan tip forme može biti označen kao *menu* ili kao *program*. Ako je tip forme označen kao *menu*, onda će taj tip forme u budućem generisanju aplikacije predstavljati osnovu za generisanje menija ekranskih ili izvještajnih formi. Ako je označen kao program, onda će taj tip forme biti osnova za generisanje ekranskih formi transakcionih programa. Detaljnije o konceptu tipa forme može se naći u [3, 4, 7].

4. KONCEPT POSLOVNE APLIKACIJE

Jednu poslovnu aplikaciju čini skup transakcionih programa namjenjenih da podrže aktivnosti na jednom tipu radnog mjesta. Način percepcije budućeg informacionog sistema od strane krajnjih korisnika bitno opredjeljuje struktura poslovnih aplikacija koje će u okviru tog informacionog sistema biti korišćene. Definisanje struktura poslovnih aplikacija predstavlja važan segment prilikom projektovanja jednog informacionog sistema.

Poslovna aplikacija je novi koncept u IIS*Case-u, putem kojeg je projektant u mogućnosti da formalno opiše funkcionalnosti informacionog sistema, vezane za uzajamne pozive transakcionih programa, odnosno generisanih ekranskih formi. Ovaj koncept definiše se na nivou aplikativnog sistema, tj. svaka poslovna aplikacija pripada tačno jednom aplikativnom sistemu. Definisanju poslovnih aplikacija mora da prethodi aktivnost kreiranja tipova formi. Tipovi formi, pored informacija o atributima i ograničenjima, nose i dodatne informacije koje se tiču transakcionih programa i njihovih ekranskih

formi. Pored ekranskih formi transakcionih programa, generator kôda će biti u stanju da produkuje i kôd koji se odnosi na međusobno pozivanje tipova formi i usklađivanje njihovih funkcionalnosti. Kada projektant ne bi imao mogućnost da definiše strukturu poslovne aplikacije tokom ove faze modelovanja informacionog sistema, tada bi morao da interveniše kreiranjem dodatnog programskog kôda koji se odnosi na organizaciju i funkcionalnost ekranskih formi generisanih transakcionih programa.

Pri kreiranju strukture jedne poslovne aplikacije, prvenstveno je neophodno definisati koji će sve tipovi formi učestvovati u toj aplikaciji. Jedan tip forme iz definisanog skupa označava se kao polazni. Ekranska forma koja je generisana na osnovu polaznog tipa forme poslovne aplikacije jeste prva forma koja će biti na raspolaganju korisniku i putem koje se može pristupiti ostalim formama.

Pored skupa tipova formi koji učestvuje u toj poslovnoj aplikaciji, potrebno je specificirati njihove međusobne veze, odnosno strukturu poslovne aplikacije. Zato je uveden novi koncept na nivou poslovne aplikacije koji se naziva *pozivajuća struktura* ili kratko *poziv*. Jedna pozivajuća struktura predstavlja uređeni par tipova formi koji učestvuju u poslovnoj aplikaciji. Ako je u okviru jedne poslovne aplikacije definisana pozivajuća strukura (A, B), tada će ekranska forma dobijena generisanjem na osnovu tipa forme A, obezbijediti poziv ekranske forme dobijene generisanjem na osnovu tipa forme B. Pri tome, A se naziva *pozivajući tip forme*, a B *pozvani tip forme*.

Tipovi formi koji učestvuju u jednoj poslovnoj aplikaciji mogu se posmatrati kao čvorovi grafa, dok se pozivajuće strukture mogu tretirati kao ivice tog grafa. Ovaj graf naziva se dijagram poslovne aplikacije i predstavlja jedan platformski nezavisan pogled na informacioni sistem. Ovaj pogled je polazna osnova za generisanje programskog kôda poslovne aplikacije, koji predstavlja novi, ovaj put platformski zavisan pogled na informacioni sistem. U cilju dobijanja što preciznijih specifikacija, konceptu pozivajuće strukture pridružene su sljedeće osobine:

– proslijeđene vrijednosti,
– način selekcije i preuzimanja podataka pri izvršenju poziva
– metod poziva i
– pozicioniranje programskih kontrola.

Ovako definisan koncept pozivajuće strukture omogućava projektantu da putem njega formalno opiše značajan dio funkcionalnosti jednog informacionog sistema koji se odnosi na povezivanje generisanih ekranskih formi transakcionih programa. U nastavku teksta, dat je detaljniji prikaz osobina pozivajuće strukture.

4.1. Proslijeđene vrijednosti

Značajnu ulogu u strukturiranju jedne poslovne aplikacije ima i definisanje podataka koji se prosljeđuju s jedne forme na drugu. U velikom broju slučajeva, ekranskoj formi potrebno je predati neki podatak da bi se ostvarila puna funkcionalnost te forme. Npr. ekranska forma koja se koristi za pregled studenata jednog departmana prilikom poziva mora dobiti informaciju o kojem se departmanu radi. U tu svrhu, tipu forme je pridružen novi koncept koji se naziva *parametar*. Na taj način, projektant ima mogućnost da definiše konkretne parametre jednog tipa forme, čija je primarna namjena čuvanje podataka koji se proslijeđuju prilikom realizacije poziva forme.

Pozivajući tip forme u neki od parametara pozvanog tipa forme može da smjesti jednu od sljedeće tri vrijednosti:

– vrijednost atributa nekog tipa komponente pozivajućeg tipa forme,
– konstantnu vrijednost ili
– vrijednost parametra pozivajućeg tipa forme.

Vrijednost parametra može biti dodijeljena nekom od atributa na pozvanom tipu forme, a takođe, kada je taj tip forme pozivajući u nekom drugom pozivu, tada vrijednost parametra može biti proslijeđena nekom parametru pozvanog tipa forme u datom pozivu.

4.2. Način selekcije i preuzimanja podataka pri izvršenju poziva

Ovaj atribut pozivajuće strukture određuje moguće ponašanje pozvane generisane ekranske forme s obzirom na selektovanje i preuzimanje podataka iz baze. U nekim slučajevima, selektovanje podataka na pozvanoj formi neophodno je vršiti samo u kontekstu vrijednosti koje su proslijeđene u okviru poziva forme. Takođe, ovaj atribut određuje da li će podaci odmah po otvaranju forme biti prezentirani korisniku. Ovom atributu moguće je pridružiti jednu od sljedeće tri vrijednosti:

– *Select on open*,
– *Restricted select* i
– *Select on open* & *Restricted select*.

Ako je atributu pridružena vrijednost *Select on open*, onda će podaci pri instanciranju pozvane generisane forme automatski biti učitani iz baze podataka i skup selektovanih podataka neće zavisiti od proslijeđenih vrijednosti. U slučaju da je vrijednost atributa *Restricted select*, tada se selektovanje podataka izvršava u kontekstu proslijeđenih vrijednosti. Kada je atributu pridružena vrijednost *Select on open & Restricted select*, tada se podaci učitavaju odmah i selektovanje se vrši u zavisnosti od proslijeđenih vrijednosti prilikom poziva.

4.3. Metod poziva

Vrijednost ovog atributa određuje međusobni odnos prozora generisanih ekranskih formi. Moguće mu je pridružiti jednu od sljedeće tri vrijednosti:

– *Modal*,
– *Non-Modal* i
– *Non-Modal* & *Close calling form*.

Ako je atributu pridružena vrijednost *Modal*, onda će generisane forme biti u *dialog* režimu rada, tj. korisnik neće biti u mogućnosti da pristupi pozivajućoj formi sve dok ne okonča

rad sa pozvanom formom. U slučaju da je vrijednost atributa *Non-Modal*, pristup pozivajućoj ekranskoj formi neće biti uslovljen stanjem pozvane ekranske forme, tj. korisnik će biti u mogućnosti da ravnopravno pristupi objema formama. Kada je atributu pridružena vrijednost *Non-Modal & Close calling form*, pozivajuća ekranska forma će biti deaktivirana i korisnik će moći da radi samo sa pozvanom formom.

4.4. Pozicioniranje programskih kontrola

Kada korisnik često pristupa pozivajućoj ekranskoj formi, tada je korisno realizovati programsku kontrolu putem koje se na brz i jednostavan način može izvršiti poziv. Vrijednost ovog atributa definiše gdje će biti pozicionirana programska kontrola putem koje će korisnik moći da izvrši poziv. Moguće je pridružiti jednu od sljedećih vrijednosti:

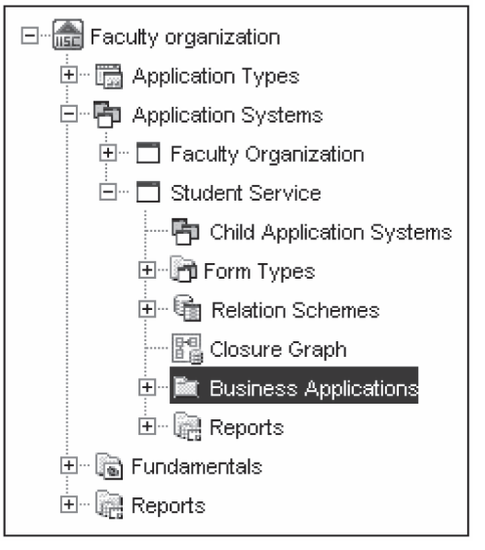
– *Show as menu item*,
– *Show as button* i
– *Show as menu item* & *Show as button*.

Ako je atributu pridružena vrijendost *Show as menu item*, tada će u meniju pozivajuće forme biti generisana stavka putem koje će korisnik biti u mogućnosti da izvrši aktiviranje pozvane forme. U slučaju da je atributu pridružena vrijednost *Show as button*, tada će u pozivajuću formu biti ugrađeno dugme putem kojeg će moći da se pristupi pozvanoj formi. Kada je atributu pridružena vrijednost *Show as menu item & Show as button*, tada će biti moguće izvršiti poziv putem obje programske kontrole.

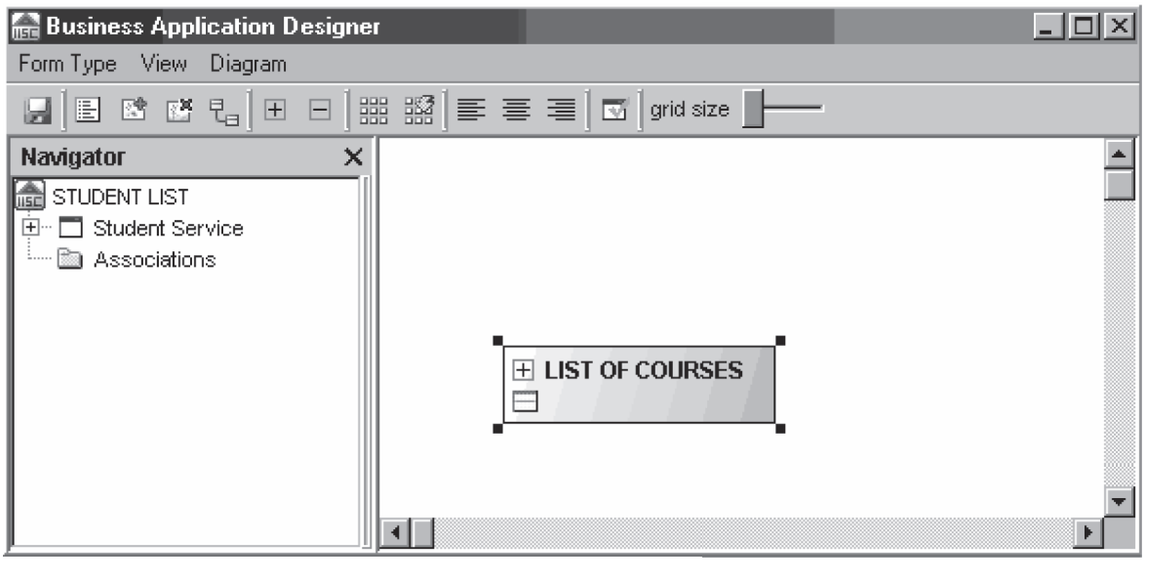
5. ALAT BUSINESS APPLICATION DESIGNER

Informacioni sistem obično obuhvata više poslovnih aplikacija, a struktura takvih aplikacija može biti različite složenosti. U cilju podrške projektovanja strukture poslovnih aplikacija, pogodno je imati alat koji će projektanu pružiti adekvatnu podršku u savladavanju složenosti poslovnih aplikacija. *Business Aplication Designer* je alat namjenjen definisanju struktura poslovnih aplikacija. Ovaj alat, ugrađen u IIS*Case, omogućava projektantu da na jednostavan i vizuelno orijentisan način specificira strukturu poslovnih aplikacija, tj. omogućava mu da definiše koji će sve tipovi formi biti uključeni u tu poslovnu aplikaciju, kao i da definiše kako se tipovi formi, tj. buduće forme aplikacije međusobno povezuju.

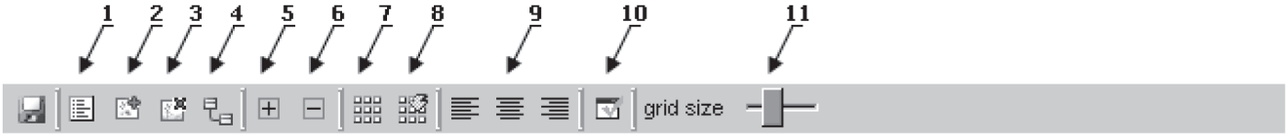
Pristup koji je prezentiran u ovom radu podrazumijeva upotrebu dijagramske reprezentacije za specificiranje struktura poslovnih aplikacija. Reprezentacija strukture poslovne aplikacije putem grafa, osim što omogućava dijagramsku, vizuelnu reprezentaciju, pogodna je jer omogućava upotrebu opšte poznatih algoritama za rad sa grafovima, kao što su algoritmi koji vrše obilazak grafa i algoritmi koji ispituju povezanost grafa. Algoritam koji ispituje povezanost grafa koristi se za provjeru validnosti poslovne aplikacije. Ako graf nije povezan, onda korisnik pojedinim ekranskim formama ne može pristupiti.



Slika 2. – *Projektno stablo za projekat Faculty organization*



Slika 3. – *Dijagram poslovne aplikacije koja sadrži samo jedan tip forme*



Slika 4. –Paleta standarnih komandi Bussines Application Designer-a

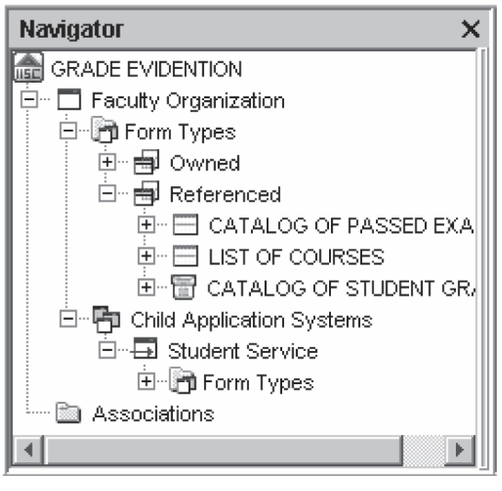
Čvoru, koji u projektnom stablu IIS*Case-a odgovara aplikativnom sistemu, podređeni su čvorovi koji reprezentuju poslovne aplikacije tog aplikativnog sistema. Na slici 2. prikazano je projektno stablo za projekat *Faculty organization*. U okviru aplikativnog sistema *Student Service* nalazi se čvor *Bussines Applications* koji reprezentuje sve poslovne aplikacije koje pripadaju tom aplikativnom sistemu.

Kreiranje poslovne aplikacije započinje zadavanjem njenog imena koje mora biti jedinstveno u okviru tog aplikativnog sistema i zadavanjem polaznog tipa forme. Po zadavanju imena i polaznog tipa forme može se pristupiti kreiranju njene strukture. Na slici 3. prikazan je izgled dijagrama najjednostavnije poslovne aplikacije koja sadrži samo jedan tip forme.

Poslije definisanja osnovnih podataka o poslovnoj aplikaciji može se pristupiti definisanju njene strukture. Projektant može da uključuje nove tipove formi, kao i da briše postojeće.

Uključivanje novog tipa forme moguće je izvršiti na više načina. Na slici 4. prikazana je paleta standardnih komandi (*toolbar*) *Business Aplication Designer*-a. Dodavanje novog tipa forme u dijagram poslovne aplikacije vrši se zadavanjem komande na koju pokazuje strelica numerisana brojem jedan. Brisanje postojećeg tipa forme iz dijagrama poslovne aplikacije vrši se zadavanjem komande na koju pokazuje strelica numerisana brojem tri.

Projektant može da odabere bilo koji tip forme iz aplikativnog sistema kojem pripada ta poslovna aplikacija, kao i tip forme koji pripada nekom od podređenih aplikativnih sistema. Takođe, moguće je uključiti i tip forme koji je referenciran od strane aplikativnog sistema kojem pripada ta poslovna aplikacija. Uključivanje novih tipova formi u poslovnu aplikaciju može se realizovati i uz pomoć navigatora. Na slici 5. prikazan je navigator *Bussines Application Designer*-a. Korisnik, ako uvidi da mu navigator nije potreban, može ga ukloniti.



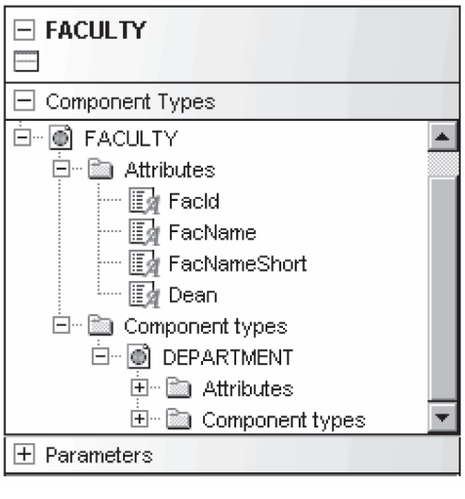
Slika 5. – *Navigator Bussines Application Designer-a*

Projektant takođe može uključivati nove tipove formi u aplikaciju prostim prevlačenjem iz navigacionog stabla. Navigaciono stablo prikazuje strukturu aplikativnog sistema kojem pripada ta poslovna aplikacija i sadrži sve tipove formi koje korisnik smije uključiti u tu poslovnu aplikaciju. Jedan tip forme može biti uključen u proizvoljan broj poslovnih aplikacija, ali ne može biti dva puta uključen u istu poslovnu aplikaciju. Ako korisnik pokuša da uključi tip forme koji je već sadržan u aplikaciji, dobiće poruku o grešci. Poslije uspješnog uključivanja tipa forme u aplikaciju, na dijagramu se pojavljuje simbol u obliku pravougaonika koji reprezentuje dati tip forme. Njegov izgled prikazan je na slici 6.

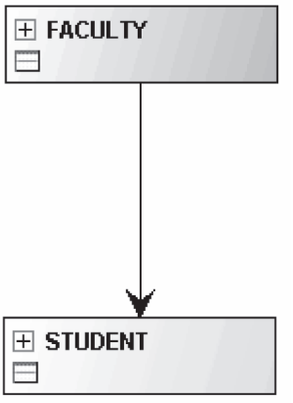
Na slici 6. prikazana je osnovna verzija simbola za tip forme. Postoji i proširena verzija, koja se dobija aktiviranjem funkije + u lijevom gornjem uglu pravougaonika. Proširena verzija simbola za prikaz tipa forme predstavljena je na slici 7. Proširena verzija obezbjeđuje pristup određenim funkcijama bitnim za definisanje strukture poslovne aplikacije, kao što su funkcije za pregled strukture tipa forme i funkcije za rad s parametrima.

Pošto projektant uključi željene tipove formi u poslovnu aplikaciju, pristupa definisanju pozivajućih struktura. Na slici 4. strelica označena brojem četiri, pokazuje na komandu putem koje se specificira poziv između dva tipa forme. U ovom režimu rada, simbole na dijagramu više nije moguće pomjerati i modifikovati, već je isključivo moguće definisati

Slika 6. – *Simbol koji reprezentuje jedan tip forme*



Slika 7. – *Prošireni simbol koji reprezentuje jedan tip forme*

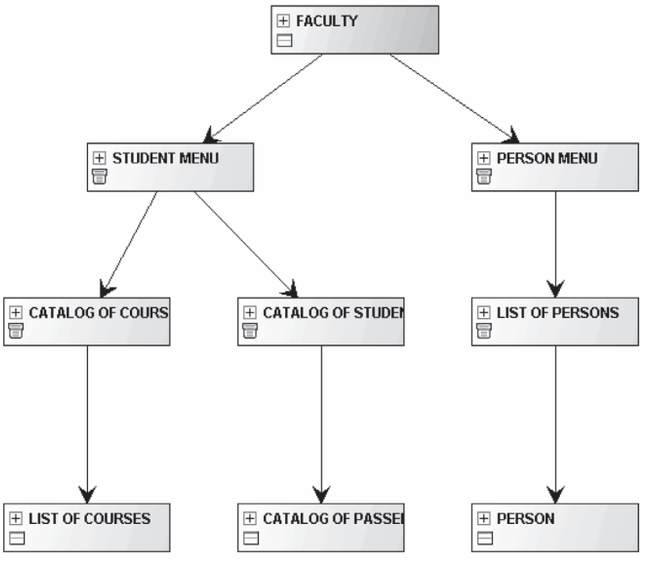


Slika 8. – *Grafički prikaz pozivajuće strukture*

veze između njih. Na primjer, ako projektant želi da kreira poziv od tipa forme *Faculty* ka tipu forme *Student*, onda to može uraditi povlačenjem linije od čvora *Faculty* ka čvoru *Student*. Na slici 8. prikazan je dijagram koji sadrži dva tipa forme i poziv između njih. Strelica koja simbolizuje poziv usmjerena je ka pozvanom tipu forme, pa se lako vidi koji tip forme je pozvan, a koji tip forme je pozivajući.

Pored definicije pozivajućeg i pozvanog tipa forme, projekatant mora da definiše i svojstva tog poziva, kako bi specifikacija pozivajuće strukture bila potpuna. Osobine poziva koje je neophodno zadati opisane su u četvrtom odjeljku ovog rada.

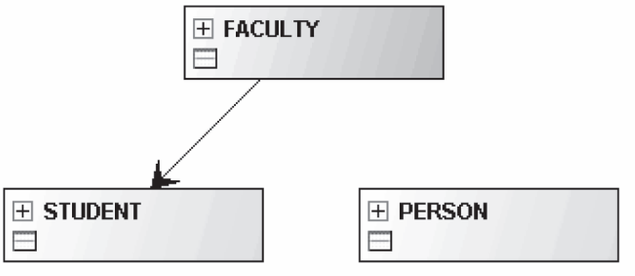
Na slici 9. prikazan je primjer dijagrama poslovne aplikacije. Vidi se da *program* tip forme *FACULTY* poziva *menu* tip forme *STUDENT MENU*. Saglasno tome, na osnovu tipa forme *STUDENT MENU* generator kôda kreiraće meni koji će biti uključen u glavni meni ekranske forme, generisane na osnovu tipa forme *FACULTY*. Takođe, sa iste slike može se uočiti da *menu* tip forme *PERSON MENU* poziva *menu* tip forme *LIST OF PERSONS*. Generator kôda će na osnovu tipa forme *LIST OF PERSONS* kreirati stavku menija koja će biti uključena u meni dobijen generisanjem na osnovu tipa forme *PERSON MENU*.



Slika 9. – *Dijagram poslovne aplikacije*

Business Application Designer pored manipulacije tipovima formi, njihovim parametrima i pozivajućim strukturama, pruža i mogućnost definisanja hijerarhijske strukture menija poslovne aplikacije. Time je postignut integrisani pristup definisanju struktura poslovnih aplikacija.

Pored ovih, projektantu su na raspolaganju i funkcije koje mu u određenoj mjeri olakšavaju rad sa dijagramom. Na slici 4, strelica označena brojem sedam pokazuje na komandu pomoću koje se aktivira mreža na dijagramu. Projektant može po želji da je prikaže ili ukloni, a takođe je moguće da, putem komande na koju pokazuje strelica pod rednim brojem osam, izvrši poravnjavanje simbola u skladu sa mrežom. Upotreba ovih fukcionalnosti olakšava rad projektantu prilikom crtanja kompleksnijih dijagrama. Korisnik je u mogućnosti da izvrši automatsko poravnjavanje simbola dijagrama u odnosu na neku od ivica prozora alata, putem grupe komandi na koje pokazuje strelica numerisana brojem devet. Simboli se mogu poravnati u odnosu na lijevu ili desnu ivicu, kao i po sredini ("centrirano"), kako bi se doprinelo preglednosti dijagrama. Prilikom projektovanja poslovne aplikacije može doći i do grešaka. Na slici 10. prikazan je primjer strukure poslovne aplikacije koja nije validna, jer se tipu forme *PERSON* ne može pristupiti ni na koji način. *Bussines Aplication Designer* vrši provjeru validnosti poslovne aplikacije. Na slici 4, strelica označena brojem deset pokazuje na komandu putem koje se vrši provjera validnosti poslovne aplikacije.



Slika 9. – *Dijagram poslovne aplikacije koja nije validna*

6. ZAKLJUČAK

Cilj ovog rada jeste da prezentira jedan metodološki pristup definisanju struktura poslovnih aplikacija. Ovakav pristup podrazumijeva postojanje posebnih koncepata i alata koji su namjenjeni u tu svrhu. Putem njih, projektant ima mogućnost integrisanog i vizuelno orijentisanog zadavanja specifikacija struktura poslovnih aplikacija, što kod drugih analiziranih alata slične namjene nije slučaj. Uvođenjem novih koncepata omogućeno je projektovanje šeme buduće baze podataka, izgleda i funkcionalnosti ekranskih formi i struktura poslovnih aplikacija na objedinjeni način, čime je u velikoj mjeri pokriven proces modelovanja softverske podrške jednog informacionog sistema. Modelovanje putem IIS*Case-a odvija se na visokom nivou apstrakcije, ali su koncepti na kojima se on zasniva povezani sa domenski specifičnim pojmovima koji se odnose na strukturu informacionih sistema. Na osnovu projek-

tovanog modela informacionog sistema, moguće je generisati implementacioni opis šeme baze podataka i programski kôd transakcionih programa. S druge strane, projektantski alati opšte namjene koji se zasnivaju na familiji jezika UML, pružaju mogućnost modelovanja široke klase sistema [12, 14], što je prednost u odnosu na IIS*Case. Nažalost, putem takvih alata teže je pri modelovanju konkretnog sistema izraziti neke specifične karakteristike, bitne za određeni domen primene, pa je dobijeni model teže transformisati u potpuno funkcionalni programski kôd u izabranom programskom okruženju.

Dalji pravci istraživanja odnose se na:

– obogaćivanje koncepata vezanih za strukturu i semantiku poslovnih aplikacija, a potom i dalje unapređivanje alata *Bussiness Application Designer*,
– izradu i unapređenje generatora kôda korisničkih aplikacija i transakcionih programa i
– proširivanje alata IIS*Case novim konceptima i programskim komponentama koje se odnose na modelovanje poslovnih procesa realnog sistema.

LITERATURA

[1] Aleksić S, Luković I, Mogin P, Govedarica M, *A Generator of SQL Schema Specifications*, Computer Science and Information Systems (ComSIS), Consortium of Faculties of Serbia and Montenegro, Belgrade, Serbia, ISSN: 1820-0214, Vol. 4, No. 2, 2007, pp. 79-98.

[2] ARTech. *DeKlarit*™(*TheModel-Driven Tool for Microsoft Visual Studio 2005*), Chicago, U.S.A. Available at: http://www.deklarit.com [March, 2008].

[3] Choobinch J., Mannio V. M., Nunamaker F. J., Konsynski R. B., *An Expert Database Design System Based on Analysis of Forms*, IEEE Transactions on Software Engineering, Vol.14, No 2, Feb. 1988, pp. 242-253.

[4] Diet J., Lochovsky F., *Interactive Specification and Integration of User Views Using Forms*, Proceedings of the Eight International Conference on Entity-Relationship Approach Toronto, Canada 18-20. October, 1989, pp.171-185

[5] Kleppe A., Warmer J., Bast W., *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison Wesley, April 2003.

[6] Koletzke P., Dorsey P., *Oracle designer handbook*, 2nd Edition, Osborne/Mc-Graw-Hill, Berkeley, California, U.S.A, 1999.

[7] Luković I., *Automatizovano generisanje podšeme relacione baze podataka putem formi*, magistarski rad, Univerzitet u Beogradu, Elektrotehnički fakultet, Smer Informatika, Beograd, 18. 06. 1993. Mentor: prof. dr Pavle Mogin.

[8] Luković I, Mogin P, Pavicević J, Ristić S, *An Approach to Developing Complex Database Schemas Using Form Types*, Software: Practice and Experience, John Wiley & Sons Inc, Hoboken, USA, ISSN: 0038-0644, DOI: 10.1002/spe.820, Vol. 37, No. 15, 2007, pp. 1621-1656.

[9] Luković I., Ristić S., Mogin P., Pavićević J., *Database Schema Integration Process – A Methodology and Aspects of Its Applying*, Novi Sad Journal of Mathematics (Formerly Review of Research, Faculty of Science, Mathematic Series), Serbia, ISSN: 1450-5444, Vol. 36, No. 1, 2006, pp 115-150.

[10] Mogin P., Luković I., Karadžić Ž., *Relational Database Schema Design and Application Generating using IIS*CASE Tool*, International Conference on Technical Informatics, Timisoara, Romania, November 16-19. 1994, Proceedings, Vol. 5, pp. 49-58.

[11] Object Management Group, *MDA Guide,* Version 1.0.1, Volume 1, document omg/03-06-01, 12th June 2003.

[12] Object Management Group, *Unified Modeling Language Specification*, Version 1.4.2, document formal/05-05-01, January 2005.

[13] Pavicević J., *Razvoj CASE alata za automatizovano projektovanje i integraciju šema baza podataka*, Magistarski rad, Univerzitet Crne Gore, Prirodno-matematički fakultet, Podgorica, 2005.

[14] Rumbaugh J., Jacobson I., *The Unified Modeling Language Reference Manual*, Addison-Wesley, U.S.A, 1999.

[15] IEEE Std 1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, Approved 21 September 2000.

Mr Aleksandar Popović, asistent, Prirodno matematički fakultet u Podgorici,

Naučne oblasti: softversko inženjerstvo, projektovanje baza podataka.

Dr Ivan Luković, redovni profesor, Fakultet tehničkih nauka u Novom Sadu.

Naučne oblasti: baze podataka, informacioni sistemi, softversko inženjerstvo.

Dr Sonja Ristić, docent, Fakultet tehničkih nauka u Novom Sadu.

Naučne oblasti: teorija i modeli baza podataka, metode i tehnike projektovanja baza podataka i informacionih sistema.