PRIMERJAVA TRADICIONALNEGA IN OBJEKTNO ORIENTIRANEGA PRISTOPA PRI RAZVOJU INFORMACIJSKIH SISTEMOV OB UPORABI ORODJA CASE

Lilijana Mihelič

Povzetek

Namen članka je predstaviti vlogo in vpliv orodij CASE (Computer Aided Software Engineering) na področju računalniške znanosti na primeru dveh orodij CASE, od katerih je v enem implementirana tradicionalna metodologija, v drugem pa objektno orientirana (v nadaljevanju OO) metodologija. Orodja nam olajšajo razvojni proces, različne vidike zajamemo s pomočjo modeliranja posameznih razvojnih faz, ki vključujejo analizo, načrtovanje in izvedbo. Primerjava vsebuje osnovne razlike objektnega razvoja v primerjavi s strukturno analizo in načrtovanjem. Medtem ko tradicionalna metoda opisuje sistem kot skupino (težko preglednih) podatkov in funkcij, OO pristop gleda na sistem kot na skupino manjših, ponovno uporabnih komponent, ki predstavljajo entitete realnega sveta.

Abstract

The article describes the role and the influence of CASE (Computer Aided Software Engineering) tools in computer science by introducing two different tools. One of them is using structured traditional development methodology while the other uses object-oriented approach. CASE tools help us to better manage the development process by modeling different views in development phases, including analysis, design and implementation. The comparison includes the basic distinction between object technology and structured analysis and design. While traditional methods are concerned with the intricacies of large amounts of data and processes, OO is concerned with reusable modules that represent real-world entities.

1. Uvod

Računalniško orodje CASE, ki bi ga pri nas lahko poimenovali računalniško podprta gradnja informacijskega sistema, je namenjeno strokovnem kadru, ki se ukvarja z razvojem novih informacijskih sistemov. V razvojnem procesu z orodji CASE lahko sledimo raznim razvojnim metodologijam, kot so naprimer SA/SD (Strukturna analiza in strukturirano načrtovanje) avtorja E. Jourdona, informacijski inženiring avtorja J. Martina, metodologija avtorjev Ward/Mellor, ki je primerna za razvoj sistemov v realnem času ali OO metodologija raznih avtorjev.

Orodja CASE, ki so bila pred desetletji pri nas skoraj nepoznana, postajajo pomembna tehnologija na področju razvoja uporabniških programov, oziroma razvoja informatike nasploh. To vlogo jim pripisujemo predvsem zato, ker omogočajo veliko hitrejši razvoj uporabniških programov ob boljši kvaliteti opravljenega dela. Danes je na tržišču prisotno večje število takih orodij CASE, ki v celoti podpirajo aktivnosti vseh faz razvojnega procesa, od analize do implementacije, zagotavljajo povezanost modelov med posameznimi fazami in povezanost med modeli znotraj ene faze in vsebujejo bogate programske pripomočke, ki na osnovi izdelanih modelov generirajo programsko kodo, namenjeno postavitvi podatkovne baze in delovanju končne aplikacije.

2. Tradicionalni razvojni pristop z uporabo orodja CASE

Kot primer takega orodja na kratko predstavljamo orodje Westmount I-CASE Yourdon, verzija 3.2, razvito v podjetju Westmount Technology B.V., ki sledi tradicionalni metodologiji avtorja Edwarda Yourdona. Kot sistem za upravljanje s podatkovno bazo uporablja Informixovo podatkovno bazo, generirana koda pa se

lahko nahaja v programskem jeziku Informix 4GL (jeziki četrte generacije), ESQL C ali C.

ESQL C (Embedded SQL for C), poimenovan tudi vstavljeni SQL C programski jezik, je jezik, ki je enak jeziku C, vanj pa so lahko s pomočjo posebne sintakse vključeni ukazi SQL, ki delajo s podatkovno bazo.

Modeli predstavljajo abstrakcijo realnosti, z njimi ponazorimo situacije realnega sveta z različnih perspektiv. V orodju lahko izdelamo tri modele: podatkovni model, procesni model in model komunikacije z uporabnikom.

Klasična metodologija postavlja mejo med podatkovnim in procesnim modelom. Nekateri avtorji pri tem pripisujejo podatkom večji pomen, kot procesom, saj naj bi bili trajnejšega značaja.

Podatkovni model zajema opis, tip in strukturo podatkov in medsebojne povezanosti podatkov, vanj so lahko vključeni tudi kontrolni podatki. Podatkovni model opisuje le, kako izgledajo podatki in ne, kaj se z njimi dogaja. Procesni model zajema opis procesov, prenos podatkov in transformacij podatkov, zajema vse, kar se dogaja s podatki. V modelu komunikacije z uporabnikom opišemo, na kakšen način bo končni uporabnik uporabljal podatke. Predvsem se to nanaša na izgled uporabniškega vmesnika, ki se realizira z zaslonskimi maskami.

3. OO razvojni pristop z uporabo orodja CASE

Pojav OO tehnologije pomeni pravi preobrat na področju računalništva. V literaturi je možno zaslediti, da ta preobrat primerjajo z revolucijo, ki so jo v elektronski industriji povzročila tiskana vezja.

Objektna tehnologija ima velik vpliv na razvoj informacijskih sistemov, kot tudi na razvoj orodij CASE. OO orodja CASE, ki omogočajo OO usmerjen razvoj informacijskih sistemov in z svojimi eksplicitnimi mehanizmi pomagajo vzpodbujati koncepte OO razvoja, so se pričela razvijati s pričetkom uvedbe OO modelirne tehnike.

Kot primer orodja, kjer je celotno razvojno okolje podprto z OO tehnologijo, predstavljamo orodje ObjectTeam, verzija 5.1.1., razvito v podjetju Cayenne Software Inc., ki omogoča razvoj projektov z uporabo objektno modelirne tehnike avtorja Jamesa Rumbaugha (v nadaljevanju samo OMT) in dodatki tehnike avtorja Ivarja Jacobsona, uporablja Informixovo podatkovno bazo, generirana koda pa se lahko nahaja v programskem jeziku ESQL C++ ali C++.

ESQL C++ (Embedded SQL for C++), poimenovan tudi vstavljeni SQL C++ programski jezik, je jezik, ki je enak jeziku C++, vanj pa so lahko s posebno sintakso vključeni ukazi SQL, ki delajo s podatkovno bazo.

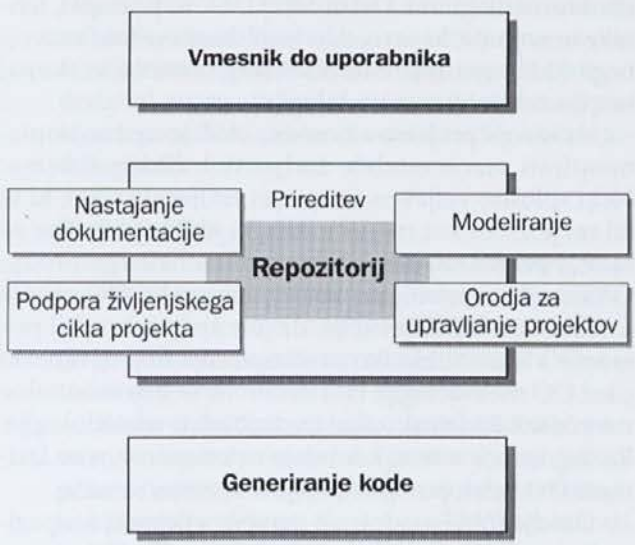
Z uporabo OMT razvijemo tri modele sistema: objektni, dinamični in funkcionalni model. Objektni model predstavlja okvir znotraj katerega lahko uspešno razvijemo še druge modele. Dinamični model opisuje aspekte modela, ki so povezani s časom in zaporedjem izvajanja operacij, opredeljuje življenjski cikel objektov in njihovo obnašanje v različnem času. Funkcionalni model zajema transformacije podatkov, opisuje od kod podatki prihajajo, kaj se z njimi dogaja in kam so poslani.

OMT poznavalci očitajo, da v nobenem modelu ne zajame opisa, ki bi predstavil, kako si objekti izmenjujejo sporočila. V ta namen OO orodje CASE ponuja možnost omenjene tri modele OMT dopolniti s komunikacijskim modelom, ki zajema opis interakcij med objekti, v funkcionalni model pa vključuje še diagram UseCase avtorja Ivarja Jacobsona.

4. Skupne značilnosti obeh orodij

S stališča funkcionalnosti, ki nam jo orodje omogoča, lahko rečemo, da orodje sestoji iz repozitorija, generatorjev dokumentacije, vmesnika do uporabnika, generatorjev programske kode in skupine grafičnih orodij, ki omogočajo modeliranje, upravljanje s projekti in podporo življenskemu ciklu projekta.

Razvojni sistem se v orodju CASE organizira kot projekt, za katerega je privzeto da se deli na faze, ki se v tradicionalnem orodju imenujejo analiza, arhitektura, načrtovanje in implementacija, v OO orodju CASE pa analiza, sistemsko načrtovanje, objektno načrtovanje in implementacija.



Slika 1: Logična zgradba orodja

Omenjeni orodji CASE sta sistema, ki delujeta na odprti računalniški arhitekturi tipa strežnik-odjemalec. Arhitektura je odprta v smislu povezljivosti in prenosljivosti na druge sisteme.

OO orodje CASE poleg že opisanega vsebuje nekatere dodatne funkcionalnosti, značilnosti in prednosti, ki jih ponuja OO tehnologija. Ena od prednosti OMT, v primerjavi s klasično metodologijo razvoja informacijskih sistemov je lažja ponovna uporaba softverskih komponent. V orodju OO CASE obstajajo orodja za povratni inženiring, ki nam omogočajo v naš projekt vključiti že razvito programsko kodo v programskem jeziku C++ ali Smaltalk. Ta koda je lahko rezultat kakega že razvitega projekta, lahko je prinešena s kakega drugega sistema ali pa je vzeta iz standardnih OO knjižnic programskega jezika C++. Orodje OO CASE prepozna dele kode, ki so ponovno uporabljeni, zato je generatorji ne spreminjajo.

5. Primerjava opisanih orodij

5.1. Življenski cikel

Življenski cikel je opis oblike razvoja sistema, ki je praviloma sestavljen iz različnih faz, ki predstavljajo različne vidike ali nivoje abstrakcije preslikave realnega sveta v aplikacijo.

Model življenskega cikla je lahko sestavljen iz podmodelov, od katerih je v vsakem zajetih več tehnik (5). V vsaki tehniki je lahko zajetih več konceptov ali osnovnih postopkov. Metodologija je množica tehnik in metod, ki so praviloma grafične, s katerimi želimo pripeljati razvoj informacijskega sistema od začetka do uspešnega konca. Primeri prvih tehnik, ki pa se uporabljajo še danes, so diagrami toka podatkov ali pa strukturni diagrami. Del orodja CASE so postopki, tehnike in metode, ki se uveljavljajo skozi vse faze razvojnega cikla izgradnje informacijskega sistema in skupaj tvorijo zaključeno metodologijo.

V razvoju projektov z orodji CASE je možno implementirati razne modele življenskih ciklov sistema. Neki splošno veljaven in sprejet življenski cikel, ki bi bil razpoznan kot najprimernejši, ni bil sprejet ne za razvoj projektov z uporabo tradicionalnega orodja CASE, niti za razvoj projektov z uporabo OO orodja CASE. Na splošno pa velja, da je v življenski cikel primerno vključiti tehniko prototipiranja in je življenski cikel OO metodologije bolj iterativne in inkrementalne narave kot življenski cikel tradicionalne metodologije. Razlog zato je v tem, ker imajo nekatere osnovne lastnosti OO pristopov ponavljajoči se rastoči značaj.

Orodja CASE podpirajo iteracije s pomočjo repozitorija, različne že izdelane prototipe ali inkremente končne aplikacije pa v orodju CASE lahko hranimo kot posebne verzije projekta.

Pri proučevanju metodologij z uporabo orodja CASE se pojavi zanimivo vprašanje: kakšen naj bi bil OO življenski cikel razvoja projekta z uporabo OO orodja CASE? Glede na teorijo in lastne izkušnje menim, da je za zajem in predstavitev OO mehanizmov v OO orodju CASE najprimernejši evolutivni življenjski cikel, v katerem je zajeto tudi prototipiranje.

V klasičnem strukturnem pristopu se srečujemo s šibko povezanostjo podatkovnega in procesnega modela, ki je zagotovljena le prek podatkovnih skladišč in podatkovnih tokov, ki se nahajajo v diagramu toka podatkov. Orodje CASE z uvedbo modela komunikacije z uporabnikom to vrzel nekoliko zmanjšuje, saj podatkovni in procesni model povezuje v diagramih za krmiljenje zaslonskih mask. Objektni pristop obravnava oba vidika v enem - v objektu, ki združuje tako podatke, kot tudi postopke, moč objektnega modeliranja in objektnega življenskega cikla se kaže v tem, da je isti objekt prisoten v vseh fazah, od opisa problema do izdelane rešitve. Razlika je le v tem, da je vloga objekta v fazi analize modelirati poslovne funkcije, medtem ko je objekt v fazi implementacije podrobneje prečiščen in dopolnjen z drugimi objekti, ki so namenjeni izvedbi. V fazi implementacije v OO orodju CASE objekt ne nastopa več v objektnih diagramih, pač pa v kodi OOPL, ki deluje nad relacijsko podatkovno bazo in s tem zmanjšuje pridobitve OO razvoja.

Zaključki mnogih avtorjev, ki so proučevali OO življenski cikel (Henderson-Sellers, Booch, Bailin, 5) so, da cikel vsebuje tako dekompozicijo funkcij s pristopom od-zgoraj-navzdol, kot tudi izgradnjo sistema po principu od-spodaj-navzgor. Rezultat analize, izvedene s pristopom izgradnje od-zgoraj-navzdol, je množica temeljito proučenih razredov, definiranih na najvišjem nivoju abstrakcije, ki vključujejo tako mehanizme dedovanja kot tudi opis sporočil, ki poteka med objekti. V fazi načrtovanja je opis razredov podrobnejši in zahteva tudi opis njihovega obnašanja in uvedbo dodatnih razredov, ki služijo namenu implementacije. Od tu dalje se v postopku modeliranja poslužujemo principa od-spodaj-navzgor, s pomočjo katerega dopolnimo obstoječe objektne diagrame na način, primeren za realizacijo. Načrtovanje sestoji iz načrtovanja razredov in načrtovanja aplikacije. Načrtovanje razredov vključuje procedure po obeh pristopih, medtem ko se pri načrtovanju aplikacije izkaže primernejši pristop od-spodaj-navzgor. Kodiranje se pogosto prične, že preden se konča načrtovanje. Tega ne preprečuje niti postopek razvoja v OO orodju CASE, saj kljub avtomatskemu generiranju (dokaj skromne kode) od razvijalca celo zahteva, da v telo funkcij, ki pripadajo nekemu razredu, vpiše programsko kodo, ki se v postopku generiranja le preslika na ustrezno mesto določene datoteke. Omenjeno dejstvo in pa prehod iz faze objektnega načrtovanja v fazo

implementacije OO orodja CASE, kjer se izvaja avtomatično generiranje kode, povzroča večjo prepletenost načrtovanja in kodiranja, kot je to prisotno v strukturiranem, tradicionalnem pristopu.

Z zajemom obeh pristopov (od-zgoraj-navzdol in od-spodaj-navzgor) v istem življenskem ciklu dosežemo bolj fleksibilno načrtovanje in boljšo interakcijo okolja s sistemom.

Slika 2 prikazuje zanko med načrtovanjem in kodiranjem, rezultat katere je lahko v vsaki iteraciji nov prototipni izdelek končne programske opreme. Ponovni vhod v fazo načrtovanja ali fazo analize je povezan z zamrznitvijo verzije pravkar izdelanega prototipa in vseh tistih delov modela, ki jih v ponovni iteraciji mislimo spreminjati.

5.2. Metodologija

Ključna razlika med tradicionalnim pristopom in objektno metodologijo je v načinu razmišljanja in gledanja na sistem. Modelirne tehnike OO razvoja se vse sklicujejo na en sam osnovni element sistema - to je razred objektov, medtem ko klasična analiza in načrtovanje modelirata sistem z različnimi osnovnimi elementi. Entitete so osnovni gradniki diagramov entiteta-povezava, aktivnosti pa so osnovni gradniki diagramov toka podatkov. To vodi v slabo povezanost med različnimi modeli sistema, kar se odraža v tem, da je težje doseči modularno zgradbo končnega sistema. OO razvoj sistemov, ki vključuje OO mehanizme (dedovanje, polimorfizem, ...), omogoča ponovno uporabo že razvitih komponent razvoja in razširljivost.

Kar se tiče notacij in diagramskih predstavitev lahko ugotovimo, da se tradicionalna in objektna metodologija bistveno ne razlikujeta. Kljub temu, da pri OO pristopu temeljimo na novem konceptu - objektu, zaznamo vpliv strukturne analize in informacijskega inženirstva. Podobnosti se kažejo v stičnih točkah, kljub temu, da OO metode vnašajo nove konstrukte in koncepte, s čimer skušajo predstaviti OO principe.

V opisanih dveh orodjih CASE je ta razlika bolj očitna, saj OO orodje CASE ponuja k vsem diagramom OMT še tri diagrame, kjer je možno ponazoriti informacije, ki jih OMT v svoje diagrame ne vključuje.

Kar se tiče analize in predvsem načrtovanja, so razlike med tradicionalnim in OO pristopom nekoliko večje. Kadar prehajamo od tradicionalne metodologije na OO metodologijo, moramo premostiti vse te razlike. Razlike so torej razvidne iz postopka prehoda med omenjenima metodologijama.

5.2.1. OO metodologija kot prehodna metodologija

Poleg tega, da predstavljajo objektno metodologijo kot razvojno metodologijo, s katero je moč v celoti razviti sistem, naj bi tudi omogočala uspešnejši razvoj v kombinaciji s tradicionalnim pristopom. Torej, objektno metodologijo je moč uporabiti kot prehodno metodologijo, ki združuje tako značilnosti OO, kot tudi značilnosti tradicionalne strukturirane metodologije.

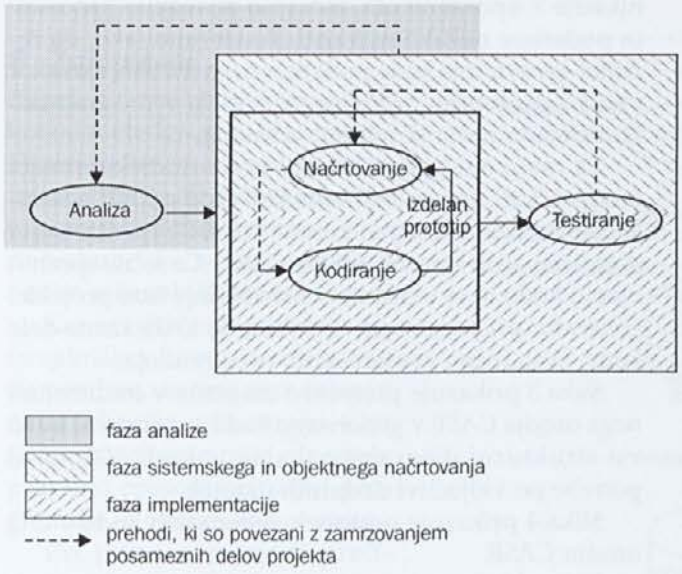
V postopku modeliranja naj bi bil ta prehod možen z uporabo OO diagramov v kombinaciji z diagrami, ki so že razviti s tradicionalnim pristopom. Primer: vse informacije, ki so zajete v diagramu entiteta-povezava, lahko predstavimo v objektnem diagramu, ki ga poleg tega dopolnimo še z značilnostmi OO mehanizmov.

Uporaba tradicionalnih metod na začetku življenskega cikla projekta ne izključuje uporabe OO metod v nadaljevanju razvoja projekta. Kombinacija tradicionalnih in OO pristopov nas lahko privede do odličnih rešitev. Prav tako je ta način primeren, ker razvijalcem ni potrebno naenkrat usvojiti obširno dodatno znanje in ker se ne srečujemo s kompleksnim procesom pretvorbe med različnimi metodologijami.

Prehod v fazi analize

Spisek zahtev in opis problemske domene, ki je izdelan z uporabo tradicionalnega orodja CASE, je povsem uporaben tudi pri OO pristopu, s tem da ga lahko še dopolnimo z značilnostmi OO metodologije.

Kadar izdelujemo analizo zelo obsežnih sistemov, je doprinos vnešenih sprememb v obstoječe procese majhen. Veliko aktivnosti prinese takorekoč identične rešitve. To pripisujemo dejstvu, da je OO pristop v fazi analize bolj osredotočen na modeliranje entitet realnega sveta (objektov), kot pa na modeliranje procesov.



Slika 2: OO življenski cikel v OO orodju CASE

Če pogledamo model, ki služi postavitvi podatkovne baze, je v primeru OO metodologije le ta predstavljen z objektnimi diagrami, medtem ko je v primeru modeliranja z uporabo tradicionalnih metod predstavljen v diagramu entiteta-povezava. Diagrami entiteta-povezava silijo razvijalce, da že v fazi analize v diagrame vnašajo odločitve, ki jih OO pristop pripisuje fazi načrtovanja. Konkretno, tradicionalno orodje CASE pri preverjanju konsistentnosti diagramov entiteta-povezava v fazi analize zahteva že vnešene atribute entitet in opredelitev tipov podatkov za atribute.

Pri OO pristopu so podatki predstavljeni kot razredi. Pod pojmom razredi so v nadaljevanju mišljeni razredi objektov. Razred združuje tako podatke, kot tudi postopke. Lahko bi rekli, da so ti isti razredi v diagramih entiteta-povezava predstavljeni tako skromno, da v fazi načrtovanja in implementacije ne predstavljajo neke smiselne predstavitve glede na aktivnosti sistema. Zato razvijalec porabi dodaten čas, da te alternativne predstavitve združi v tako obliko, da je smiselna in semantično koristna za nadaljni razvoj. Za OO pristop pa velja, da so podatki, ki so predstavljeni kot razredi v objektnih diagramih, že zajeti na tak način. V kolikor pa ne soupadajo z izdelavo projekta, je iz kode za postavitev podatkovne baze, ki jo dobimo kot rezultat objektnih diagramov, s pomočjo povratnega inženiringa možno dobiti željene diagrame entiteta-povezava.

Za izdelavo procesnega modela, ki je narejen z uporabo tradicionalnih pristopov, velja, da ga je bolje dopolniti z OO mehanizmi, kot nadomestiti z novim modelom. To je povsem zadovoljivo, saj je namen analize zajeti in opisati zahteve in ni nujno, da smo pri tem striktno privrženi le eni razvojni tehnologiji.

Prehod v fazi načrtovanja

Nekatera izhodišča, ki so opisana v fazi načrtovanja, so pravzaprav zahteve, ki so pogosto zajete že v fazi analize. To je naprimer potreba po porazdeljenem sistemu ali kompatibilnost z določenim operacijskim sistemom.

Sistemsko načrtovanje opredeljuje stvari, ki so povezane s funkcionalnostjo ali fizičnimi lokacijami, izvedba teh opisov pa je lahko realizirana z različnimi tehnikami in orodji. Zato je v fazi sistemskega načrtovanja najlaže določiti, kateri podsistem bo nadalje razvit z uporabo OO metodologije.

Faza objektnega načrtovanja je faza, kjer se najpogosteje prične izvajati prehod na OO metodologijo. Prehod je tu najmanj kritičen, hkrati pa je še vedno možno zajeti in vključiti vse prednosti OO metodologije. Izstopa načrtovanje ponovno uporabnih komponent, moduli so manj obsežni, možno jih je nadomestiti ali ponovno uporabiti.

Prehod med tradicionalno analizo in OO načrtovanjem zahteva preslikavo entitet, ki zajemajo le opis podatkov, v podatkovni del razreda in nekoliko težjo preslikavo diagrama toka podatkov v opis funkcij razreda. Pri tej preslikavi se lahko zgodi, da tvorimo razrede, ki nimajo svojega obnašanja ali pa razrede, ki nimajo podatkovnega dela. To je posledica slabe povezanosti med procesnim in podatkovnim modelom strukturiranega modeliranja. V takem primeru je potrebno ponovno analizirati celoten sistem, z upoštevanjem nove tehnologije. Nato združimo določene razrede in vpeljemo še dodatne, nove razrede. Na ta način si podvojimo nekoliko dela, saj smo poleg izdelane klasične analize izdelali še del OO analize.

5.3. Generatorji kode

Ker se tudi sama ukvarjam s pisanjem programske kode, je zame zelo zanimiv podatek, na kakšen način in kakšno programsko kodo lahko dobimo s pomočjo generatorjev omenjenih dveh orodij.

Medtem, ko je v tradicionalnem orodju CASE postopek pretvorbe diagramov v programsko kodo sestavljen iz treh korakov, ki jih moramo izvesti ročno, je v OO orodju CASE postopek avtomatičen in se izvede ob prehodu v fazo implementacije.

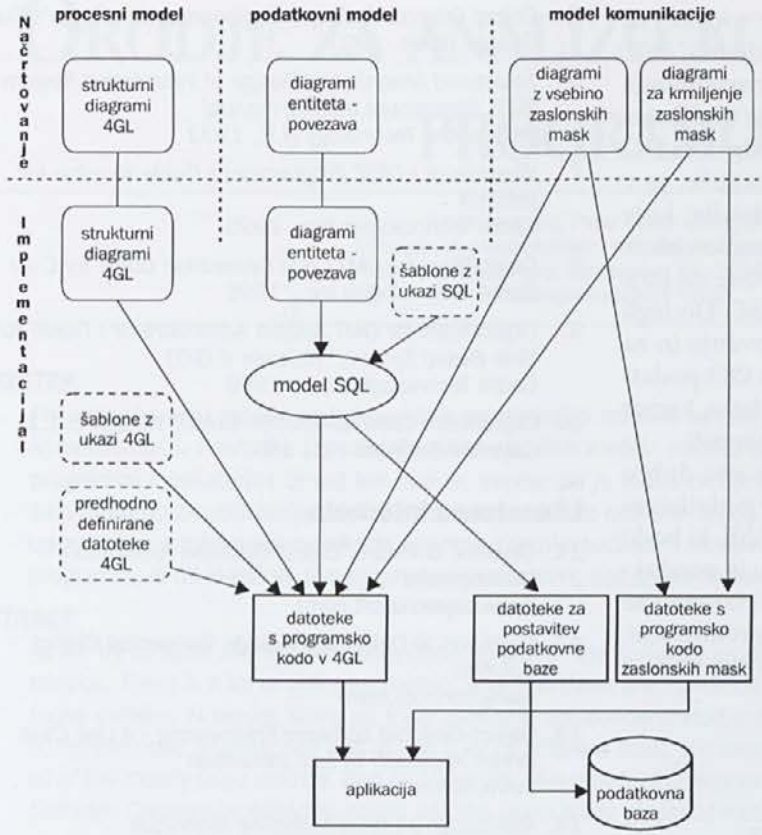
Postopek v OO orodju CASE je prijaznejši in enostavnejši. Vzrok za to je v tem, da je to orodje nastalo kasneje in je zato v tehničnem in funkcionalnem pogledu neka nadgradnja tradicionalnega orodja CASE. Vendar pa, če bi tradicionalno orodje skušali še tako izpopolniti, se srečamo z omejitvami, ki se nanašajo na modelirno tehniko. Kar se tiče tradicionalne razvojne metodologije, se zopet srečujemo s problemom ločenih modelov; med procesnim in podatkovnim modelom namreč ni prave povezave. Kljub temu, da orodje CASE skuša to vrzel zapolniti z izdelavo modela komunikacije z uporabnikom, moramo povezavo procesov in podatkov nekako ustvariti. To storimo tako, da rezultat generiranja kode podatkovnega modela služi kot vhod v pretvorbo procesnega modela v programsko kodo. Na ta način se modela združita.

Za razliko od generatorjev kode tradicionalnega orodja CASE, generatorji kode OO orodja CASE vsebujejo mehanizem za regeneriranje kode. Če spreminjamo diagrame, se to sprotno odraža v kodi. Če ročno spreminjamo kodo in se vračamo v predhodnje faze projekta, generator pri ponovnem generiranju kode zazna dele kode, ki so ročno vnešeni in jih ne spreminja.

Slika 3 prikazuje pretvorbo diagramov tradicionalnega orodja CASE v generirano kodo, v primeru, ko se vsi strukturni diagrami nahajajo v kodi 4GL in ni potrebe po vključitvi dodatnih datotek.

Slika 4 prikazuje postopek generiranja kode v OO orodju CASE.

Ko preidemo iz faze objektnega načrtovanja v fazo implementacije, generator kode avtomatično spremeni diagrame objektnega modela v datoteke s programsko



Slika 3: Prikaz povezave diagramov in postopka generiranja kode tradicionalnega orodja CASE. (7)

kodo. Generatorji omenjenega OO orodja CASE generirajo kodo aplikacije v OO programskem jeziku C++ in kodo z ukazi SQL za postavitev podatkovne baze iz objektnih diagramov.

6. Zaključek

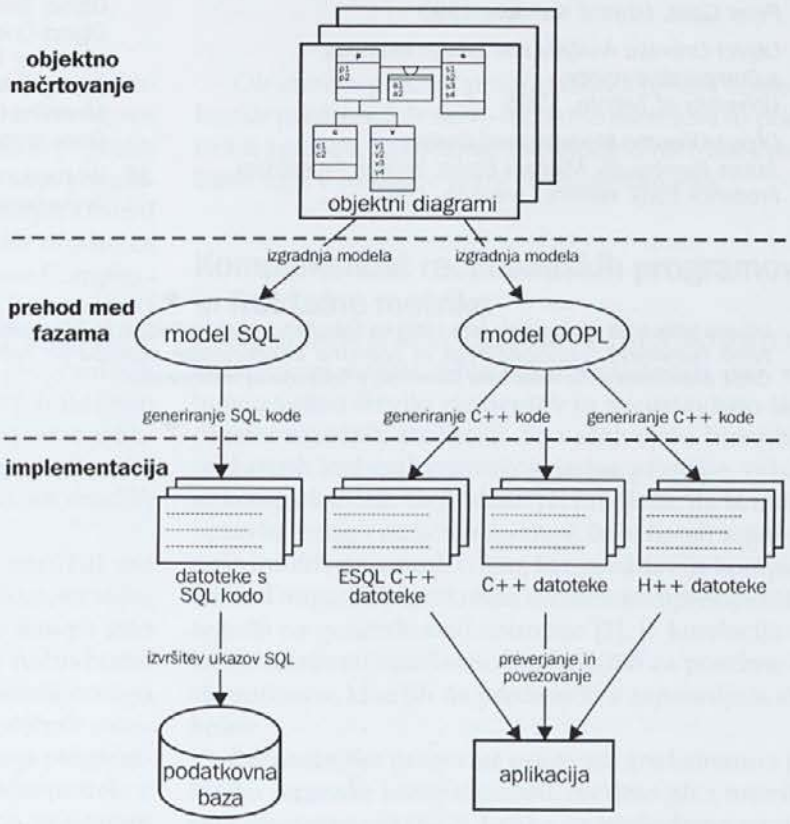
Uspešen razvoj projektov je bolj kot od metodologije, ki jo uporabljamo, odvisen od znanja, ki ga imamo. Poleg tehničnega znanja je pomembna tudi prožnost, ki je potrebna, da se prilagajamo novim računalniškim orodjem.

Ker se tržišče na področju računalništva hitro spreminja, je pomembno, da smo seznanjeni z najsodobnejšo tehnologijo. Čim širše poznavanje nam omogoča pravilno presojo pri izbiri programske in aparaturne opreme.

Pri tržno usmerjeni predstavitvi novih produktov je moč opaziti veliko pretiravanje v naštevanju prednosti, ki naj bi jih novi produkti prinesli. Bolj ko smo nepristranski in več ko imamo znanja in izkušenj z uporabo različnih obstoječih računalniških produktov, bolj realno in pravilno znamo ovrednotiti dejanske prednosti novih produktov in v zvezi z njimi tudi morebitne pomanjkljivosti.

Pojav OO metodologije se na računalniškem tržišču predstavlja kot fenomen, ki zagotavlja uspešnejši razvoj novih informacijskih sistemov. Uspeh naj bi bil zagotovljen zaradi boljšega in lažjega modeliranja sistema, ponovne uporabnosti in jasnosti problema, kar naj bi zagotavljal pogled na realni svet s pomočjo objektov.

OO orodje CASE pomaga spodbujati koncepte OO razvoja. Kljub temu, da poleg OO modeliranja vključuje vse pridobitve OO razvoja, kot je naprimer podpora ponovne uporabe,



Slika 4: Prikaz povezave diagramov in postopka generiranja kode OO orodja CASE (10)

ugotavljamo, da glede na nekatere vidike ni nikakršnega doprinosa v primerjavi z orodjem CASE, ki sledi tradicionalni metodi razvoja informacijskega sistema. Razlog za to je v tem, da OO orodje CASE razvojno in produkcijsko okolje ni v celoti podprlo z OO metodologijo. Primer, kjer se to jasno izkaže, je v postavitvi podatkovne baze končne aplikacije, ki je praktično enaka, ne glede na to, ali modeliramo sistem z uporabo tradicionalne metode orodja CASE, ali pa z uporabo OO modeliranja OO orodja CASE. Do tega pride, ker OO orodje CASE za svoje delovanje in za delovanje razvitih projektov ne uporablja OO podatkovne baze, pač pa le relacijske podatkovne baze, kamor je težko vgraditi koncepte objektne orientiranosti.

V prihodnosti lahko pričakujemo vse več dobro testiranih in komercialno uveljavljenih OO podatkovni baz, kar bo očitno tudi v razvoju orodij CASE, ki bodo z OO tehnologijo podprla celotno razvojno in produkcijsko okolje. Na ta način bodo prišle do izraza vse prednosti OO tehnologije, skupaj z vsemi prednostmi, ki jih ponuja razvoj novih informacijskih sistemov z orodji CASE.

Literatura

1. *Načrtovanje in gradnja informacijskih sistemov*
   Andrej Kovačič, Mirko Vintar, 1994
2. *Object-Oriented Analysis*
   Peter Coad, Edward Yourdon, 1990
3. *Object-Oriented Analysis and Design Methods,*
   a Comparative review,
   University of Twente, 1995
4. *Object-Oriented Modeling and Design,*
   James Rumbaugh, Michael Blaha, William Premerlani,
   Frederick Eddy, William Lorensen, 1991
5. *Object-Oriented Software Engineering, Developer's Guide*
   George Wilkie, 1993
6. *Structured Analysis and Design of Information System with ISEE, Westmount training manual*
   Westmount Technology B.V., 1993
7. *Westmount I-CASE Programmer's Guide Yourdon for Informix*
   Cadre Technologies Inc., 1995
8. *ObjectTeam for OMT Code Generation Guide for C++*
   Cadre Technologies Inc., 1996
9. *ObjectTeam for OMT System Administrator's Guide for Unix-Based System, Release 4.0/01*
   Cadre Technologies Inc., 1996
10. *ObjectTeam Code Generation Guide, Version 5.1.1.*
    Cayenne Software Inc., 1997

Literatura z Interneta

11. *Cayenne Software, Object-Oriented Application Development*
    (www.cayennesoft.com)
12. *Migration to Object technology, Connected Object Solution*
    (www.connobj.com)
13. *Object-Oriented Software Engineering - A Use Case Driven Approach by Ivar Jackobson*
    (www.rational.com)
14. *Rumbaugh's Object Modeling Technique*
    (www.cgi.com)
15. *The Object Advantage-Business Reengineering with Object Technology,*
    Object-Oriented Analysis and Design Using the Object Modeling Technique,
    Object-Oriented Analysis and Design Using the Unified Modeling Language
    (www.rational.com)
16. *Westmount I-CASE Yourdon, Westmount Technology B.V.*
    (www.qucis.queensu.ca)

◆

Lilijana Mihelič je diplomirala leta 1990 na Fakulteti za računalništvo in informatiko v Ljubljani, kjer pripravlja magistrsko nalogo na temo Primerjava tradicionalnega in objektno orientiranega pristopa pri razvoju informacijskih sistemov ob uporabi orodja CASE.Zaposlena je na Telekomu Slovenije, v Sektorju za informatiko.

◆