UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

D I P L O M S K O D E L O

ALEKS ABRAMOVIČ

UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

D I P L O M S K O D E L O

PRIMERJAVA ORODIJ ZA RAZVOJ ZASLONSKIH OBRAZCEV: PRIMER ORACLE FORM BUILDER IN ORACLE DESIGNER

Ljubljana, avgust 2002 ALEKS ABRAMOVIČ

IZJAVA

Študent/ka _____ izjavljam, da sem avtor/ica tega diplomskega dela, ki sem ga napisal/a pod mentorstvom

_____, in dovolim objavo diplomskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne _____ Podpis

_____

Kazalo

Seznam slik

1 UVOD

Dandanes smo priče nenehnim spremembam na področju razvoja programskih orodij in informacijskih tehnologij. Računalniki in računalniška oprema so vedno zmogljivejši, dostopnejši ter nenazadnje tudi prijaznejši do uporabnika. V poslovnem svetu takorekoč ni več podjetja, ki ne bi imelo vsaj zametka podpore informacijskemu sistemu (dostop do Interneta, elektronska pošta, idr.), vse več pa jih ima že računalniško podprte vsaj strateško najpomembnejše poslovne funkcije in procese. Tako računalniška podpora sploh ne predstavlja več strateške prednosti za podjetja, temveč predstavlja nekaj, kar enostavno moraš imeti. Strateška prednost pomeni le boljšo in učinkovitejšo podporo poslovanju.

Kaj to pomeni za podjetja, ki se ukvarjajo z razvojem programske opreme? Na eni strani nedvomno zaslužek, na drugi strani pa ima hiter razvoj računalništva posledice tudi na razvijalska podjetja sama. Spreminjajo se metode in tehnike gradnje in načrtovanja informacijskih sistemov, operacijski sistemi, programski jeziki, sistemi za upravljanje z bazami, razvojna okolja in še kaj bi lahko našteli.

Razvijalska podjetja, ki želijo ohraniti korak s časom, morajo nenehno spremljati razvoj novih programskih orodij in informacijskih tehnologij. Tako je potrebno poleg menjave strojne opreme v pravem trenutku menjati tudi programsko opremo, iz česar sledi, da se je potrebno nove programe naučiti uporabljati. Zato je za razvijalska podjetja strateškega pomena izbira pravega orodja za razvoj aplikacij. Napačna odločitev bi lahko za tako podjetje pomenila obsodbo na propad oz. boj za obstanek (večmesečno izobraževanje, zunanji svetovalci, izgubljeni potencialni projekti, itn).

Na trgu obstaja velika izbira razvojnih orodij, kar nas lahko pripelje do velike zmede. Pri odločanju o nakupu programske opreme so nam lahko v pomoč določeni kriteriji, na podlagi katerih izberemo podjetju primerno razvojno orodje. Med njimi so najbolj pogosti: povpraševanje na trgu, cena samega razvojnega okolja, kakšno ceno bo imela razvita aplikacija, ustreznost zahtevam standarda ISO9001, hitrost in stroški razvoja aplikacij, stroški izobraževanja kadrov, perspektivnost orodja, priporočila svetovalnih skupin, izkušnje drugih podjetij ter ostali kriteriji.
Seveda vsi kriteriji niso enako pomembni. Kateri je za podjetje pomembnejši, je odvisno od okolja in časa. Uteži posameznih se največkrat določijo na podlagi preteklih izkušenj.

V diplomskem delu bom interes zožil na predstavitev in primerjavo dveh Oraclovih orodij, to je Form Builder-ja kot del Oracle Developer-ja in Forms Generator-ja kot del Oracle Designer-ja. V naslovu diplomskega dela nisem izrecno navedel Forms Generator-ja, ker le-ta ne predstavlja fizično ločene komponente orodja, kot je to Form Builder pri Oracle Developer-ju, temveč je integriran v sam Oracle Designer.

Namen diplome je prikazati osnovne značilnosti orodij in skozi primerjavo izpostaviti njune prednosti in slabosti. Obe sta namenjeni izgradnji zaslonskih obrazcev, razlikujeta pa se v načinu kako pridemo do končnega rezultata. Čeprav se orodji nenehno razvijata in dopolnjujeta, ostaja njun osnovni koncept nespremenjen. Izhajajoč iz tega, bo morda primerjava bralcu omogočila lažjo predstavo strukture orodij in pomagala pri odločitvi, katero orodje bi lahko bilo primernejše za lasten razvoj aplikacij.

2 PREDSTAVITEV RAZVOJNIH ORODIJ

V diplomi predstavljeni orodji za razvoj zaslonskih obrazcev sta izdelka istega podjetja, Oracle Corporation, kar je razvidno že iz samih imen. Orodji sta si v določenih pogledih in gradnikih zelo podobni, njuni produkti pa so prenosljivi med obema orodjema. Tako lahko npr. zaslonski obrazec narejen v Oracle Designer-ju brez večjih težav prenesemo v okolje Oracle Developer. V nasprotno smer, iz Oracle Developer okolja v Oracle Designer, pa je prenos nekoliko omejen, vendar izvedljiv. Prenosljivost je mogoča, ker Oracle Designer dejansko generira zaslonske obrazce za Oracle Developer okolje.
Zaradi omenjene podobnosti se bom poskušal izogniti podvajanju opisovanja objektov, ki so skupni obema orodjema. Pri tem mislim na osnovne gradnike zaslonskih obrazcev, kot so podatkovni bloki, postavke, okna in druge, ki se morda razlikujejo v poimenovanju in v nekaterih drugih lastnostih, ki so odraz načina razvoja aplikacij. Vse objekte bom natančneje opisal pri Oracle Form Builder-ju, večino teh objektov pa najdemo tudi v Oracle Designer-ju.

2.1 Oracle Developer

Oracle Developer predstavlja integrirano skupino orodij, ki omogočajo razvoj kompleksnih sistemov z uporabo grafičnih vmesnikov, podatkovnih baz, arhitektur odjemalec/strežnik in tehnologij svetovnega spleta. Sestavljen je iz zbirke orodij, ki omogočajo razvoj poslovnih sistemov (Cinarkaya, 1998):

- Oracle Form Builder (oblikovalec zaslonskih obrazcev),
- Oracle Report Builder (oblikovalec poročil),
- Oracle Graphics Builder (oblikovalec grafikonov),
- Oracle Procedure Builder (oblikovalec procedur),
- Oracle Project Builder (oblikovalec, organizator projektov),
- Oracle Query Builder (oblikovalec poizvedb),
- Oracle Schema Builder (oblikovalec baznih objektov),
- Oracle Translation Builder (prevajalnik aplikacij).

Oracle Developer omogoča kreiranje aplikacij na podlagi baznih definicij brez pisanja ene same programske vrstice. Omogoča takojšen dostop do baze podatkov za branje in pisanje, ki je brez programskih napak in ga ni potrebno vzdrževati. Uporabniški vmesnik okolja Oracle Developer sestavljajo vsestransko uporabna in enostavna kombinacija iskalnikov objektov, palete lastnosti, drevesna struktura ter zmogljivi čarovniki. Prav tako orodje podpira možnost ponovne uporabe komponent iz predlog in objektnih knjižnic, kar lahko bistveno prispeva k večji produktivnosti razvijalcev. V svojem vmesniku združuje vse prednosti predmetno usmerjenega programiranja, ne da bi od razvijalcev zahtevalo dolgo dobo učenja.
Z uporabo komponente Oracle Project Builder nam Oracle Developer ponuja razvojno okolje za vse velikosti skupin in projektov, upravlja in shranjuje vse vrste aplikacij in druge zunanje komponente ter omogoča njihovo zaganjanje v izbranem orodju.

Povezavo Oracle Developer z drugimi aplikacijami in orodji je omogočeno skozi OCX/ActiveX kontrole, OLE objektov (ang. Objekt Linking and Embedding) in DDE (ang. Dynamic Data Exchange). Podpiranje široke palete multimedijskih formatov je dopolnjeno z odprtim programskim vmesnikom razvojnega orodja (ang. API – Application Programming Interface), kar omogoča večjo fleksibilnost pri podaljševanju aplikacij in pri integriranju drugih komponent v njihovo funkcionalnost. Poleg popolne integracije z Oraclovimi bazami omogoča transparentno povezavo z vsemi večjimi bazami kot so Rdb, SQL Server, Informix, Sybase, DB2 in drugimi (Oracle Developer/2000, str. 3).

V nadaljevanju se bom v skladu z naslovom diplomskega dela osredotočil na orodje za razvoj zaslonskih obrazcev, to je Oracle Form Builder.

2.1.1 Oracle Form Builder

Oracle Form Builder je zmogljivo RAD[1] orodje za razvoj uporabniku prijaznih zaslonskih obrazcev, ki lahko delujejo tako v načinu odjemalec – strežnik (ang. client – server) kot tudi preko Interneta[2]. Tako narejeni zaslonski obrazci omogočajo enostavno pregledovanje, spreminjanje, dodajanje in brisanje podatkov v podatkovni bazi. Primerni so za vse nivoje in potrebe poslovnih sistemov, od najmanjših z nekaj uporabniki pa do takih z nekaj 1000 uporabniki. Pri razvoju zaslonskih obrazcev sodelujejo tri komponente Form Builder-ja (Inroduction to Oracle Forms, str. 2):

- Oracle Forms Designer
- Oracle Forms Generate
- Oracle Forms Runtime

Oracle Forms Designer predstavlja osnovno razvojno orodje v katerem razvijalec oblikuje programsko rešitev. Ta zajema same zaslonske obrazce, menuje, PL/SQL knjižnice in druge. Zaslonske obrazce je možno oblikovati ročno oz. s pomočjo čarovnika (ang. wizard), ko gre za enostavnejše rešitve. Organiziran je v drevesno strukturo, ki daje orodju preglednost in pričakovano organiziranost. Uporabniku je na voljo je skupek pogovornih oken, ki omogočajo kreiranje različnih objektov, definiranje njihovih lastnosti in pisanje programske logike.

---

[1] (ang. Rapid Application Development). Metodologija, ki omogoča razvijalcem hiter razvoj delujočih aplikacij. V splošnem RAD sistemi ponujajo orodja za pomoč pri oblikovanju grafičnih vmesnikov, za katere bi sicer potrebovali več razvojnega napora in časa.
Bistvena razlika med RAD pristopom in klasično tehniko programiranja je v tem, da nam pri RAD pristopu ni potrebno celotnega programa kodirati, ampak imamo že narejene določene komponente, ki jih grafično vstavljamo v program (program sestavljamo), programirat pa moramo samo akcije nad določenimi komponentami.

[2] Orodje omogoča izdelavo Java aplikacij za pregledovanje, spreminjanje, dodajanje in brisanje podatkov v podatkovni bazi, ne da bi bilo treba napisati eno samo vrstico v Javi.

Komponenta Oracle Forms Generate omogoča kreiranje izvedbene (zagonske) datoteke, ki se bo uporabljala v delovnem okolju. Tako narejena datoteka se razlikuje od tiste, ki smo jo izdelali v Oracle Forms Designer-ju, in sicer se njene vsebine ne da več popravljati in pregledovati. Pri zaslonskih obrazcih dobi ta datoteka končnico *.fmx, medtem ko ima izvorna datoteka v času razvijanja aplikacije končnico *.fmb. Med generiranjem izvedbenih datotek se prevede oz. pregleda pravilnost sintakse programske logike aplikacije in druge morebitne napake. Ob pojavu kakršne koli napake se izvedbena datoteka ne zgenerira, orodje pa nas obvesti o tipu in delu aplikacije, kjer se je napaka pojavila.

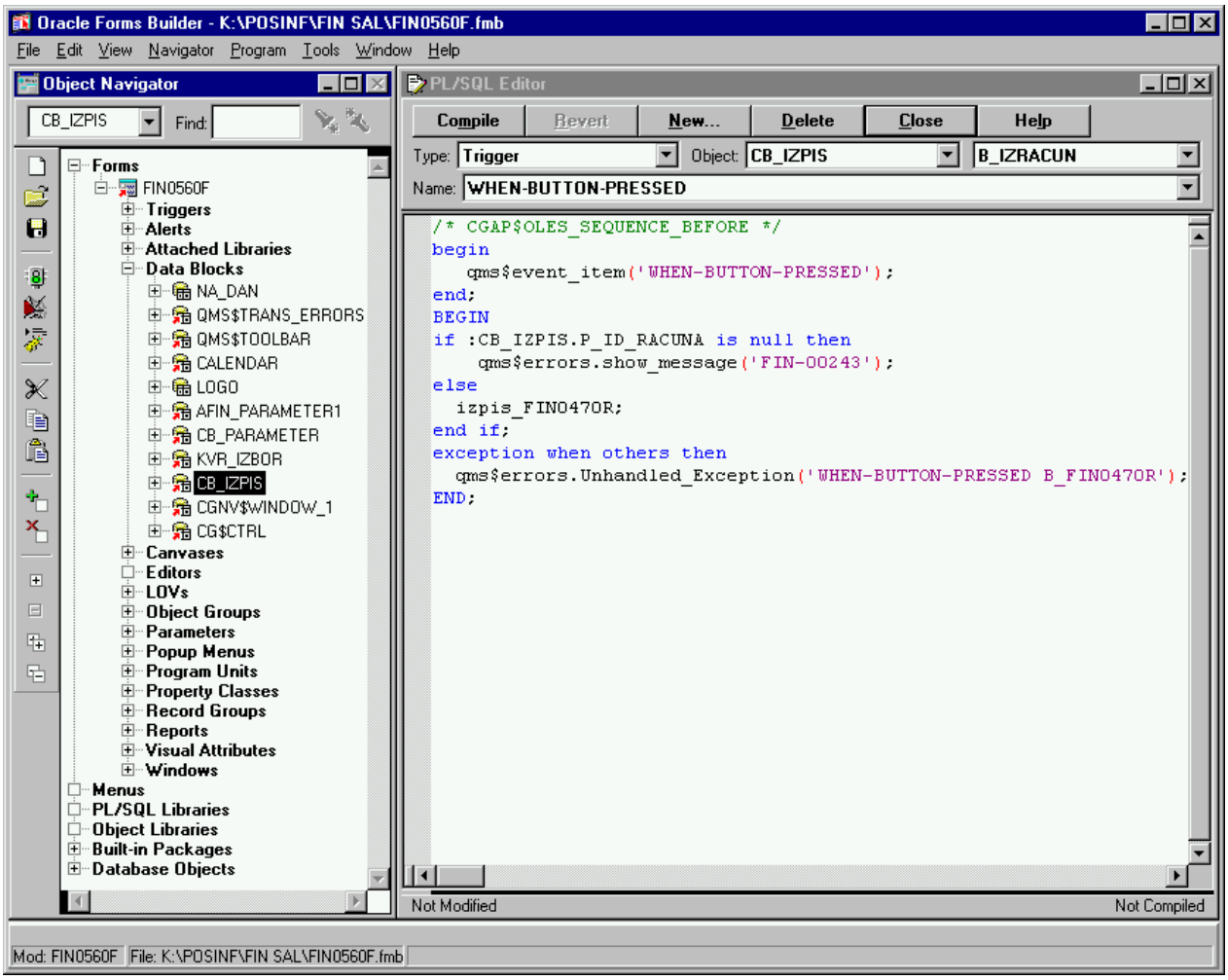
Oracle Forms Runtime predstavlja tisti del Oracle Form Builder-ja, ki omogoča zaganjanje izvedbenih datotek.

Grafični vmesnik orodja sestavljajo objektni navigator (ang. Object Navigator), paleta lastnosti (ang. Property Palette), integriran PL/SQL urejevalnik (ang. PL/SQL Editor) in urejevalnik izgleda (ang. Layout Editor/Layout Model). Predstavljajo osnovna pogovorna okna, ki jih uporabljamo pri kreiranju aplikacij in njihovih gradnikov. Najdemo jih v menuju Orodja.

2.1.1.1 Objektni navigator

V osrčju razvojnega okolja je objektni navigator, lahko prepoznaven vmesnik za pregledovanje in urejanje gradnikov aplikacije, ki omogoča razvijalcem preglednost in enostavno upravljanje z objekti. Urejen je v drevesno strukturo, kjer so posamezne veje rezervirane za določene vrste objektov. S tem postane razvoj aplikacij enostavnejši, novim razvijalcem pa omogoča lažje razumevanje razvoja programskih rešitev. Če želimo ponovno uporabiti predmete ali razrede predmetov, znotraj programov ali med njimi, ali seveda med odjemalcem in strežnikom, zadostuje že preprosta operacija povleci – spusti (ang. Drag & Drop). Objekti so urejeni hierarhično, na najvišjem nivoju pa najdemo objekte v sledečem vrstnem redu: forme, menuji, knjižnice, vgrajeni paketi in objekti baze podatkov, ki pa so vidni samo takrat, ko smo povezani na podatkovno bazo.

Slika 1: Objektni navigator



Vir: Oracle Form Builder, 2002.

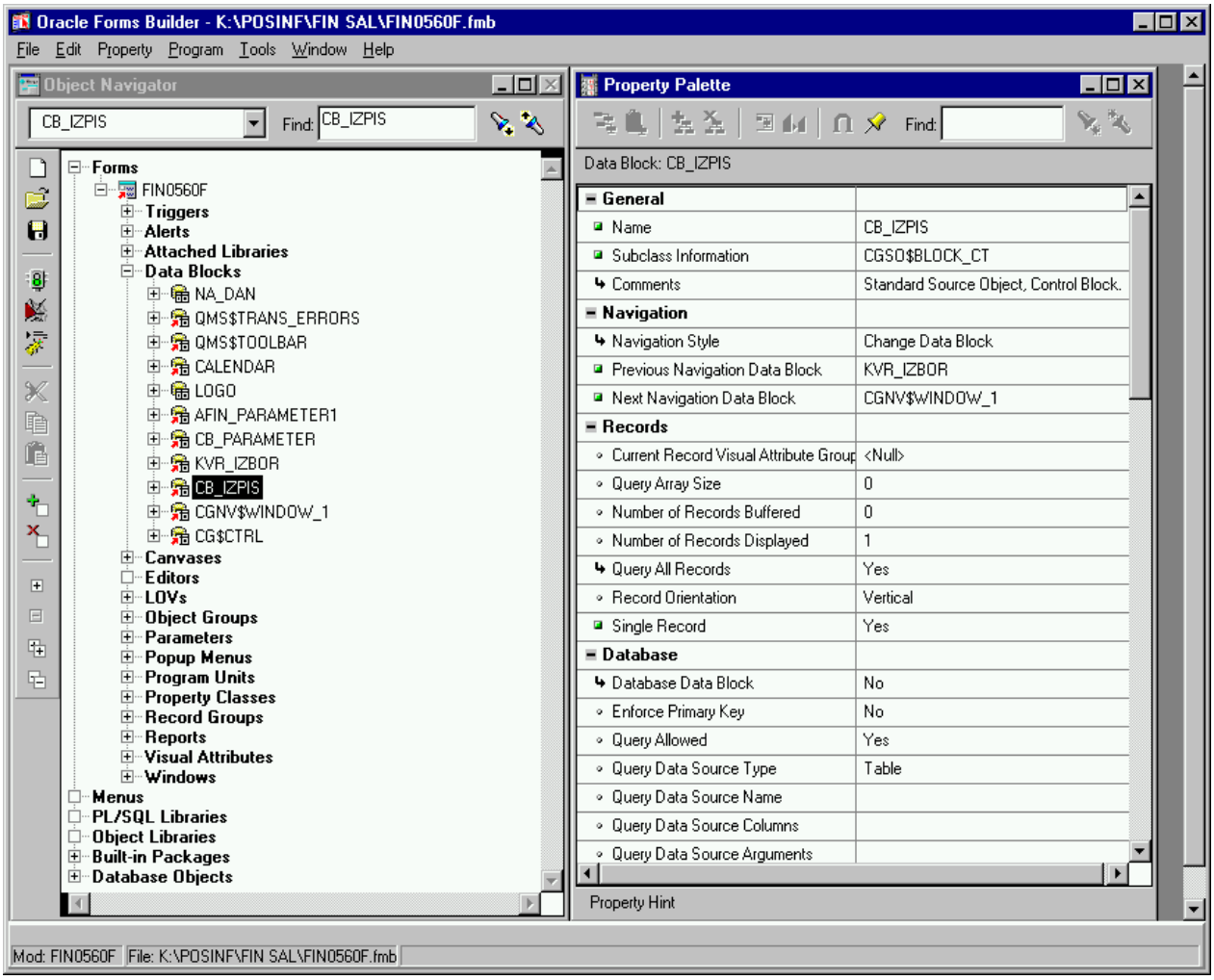
V levem delu okna (glej sliko 1, na str. 6) je prikazan objektni navigator, na njegovi desni pa urejevalnik programske logike, kateri omogoča pisanje, urejanje, brisanje in preverjanje pravilnosti sintakse programske logike.

2.1.1.2 Paleta lastnosti

Vsi objekti forme, vključno s formo samo, imajo lastnosti, katere lahko vidimo in prirejamo v pogovornem oknu za lastnosti. Vsak objekt dobi ob kreiranju privzete lastnosti, ki jih lahko po potrebi poljubno spreminjamo. Lastnosti je možno kopirati iz drugih objektov, objekt pa jih lahko tudi deduje od drugih objektov. Spreminjamo jih lahko tudi programsko med izvajanjem programske logike, kar pomeni, da se lastnosti dinamično spreminjajo med samim izvajanjem aplikacije.
Različni tipi objektov imajo različne lastnosti. Tako imajo npr. postavke lastnosti kot so: ime, širina, višina, tip podatka, ki ga bo postavka predstavljala, pozicija in druge. Lastnosti, ki jih prirejamo posameznim objektom, ne vplivajo le na izgled, temveč tudi na funkcionalnost

objektov in same aplikacije. Ta način razvoja aplikacij omogoča hiter razvoj in zmanjšuje potrebo po pisanju programske logike za standardne operacije kot so:
poizvedbe, vstavljanje, spreminjanje in brisanje zapisov, definiranje poizvedb, koordinacija glavnega in podrejenega (ang. master – detail) bloka, kontrola navigacije med objekti, prikaz objektov in drugo.

Slika 2: Paleta lastnosti



Vir: Oracle Form Builder, 2002.

Na desni strani slike (glej sliko 2, na str. 7) so prikazane lastnosti podatkovnega bloka CB_IZPIS, katerega smo izbrali v objektnem navigatorju. Iz palete lastnosti lahko razberemo, da so le-te organizirane v vsebinske sklope (odebeljen tekst), ki pokrivajo različna področja lastnosti objekta.

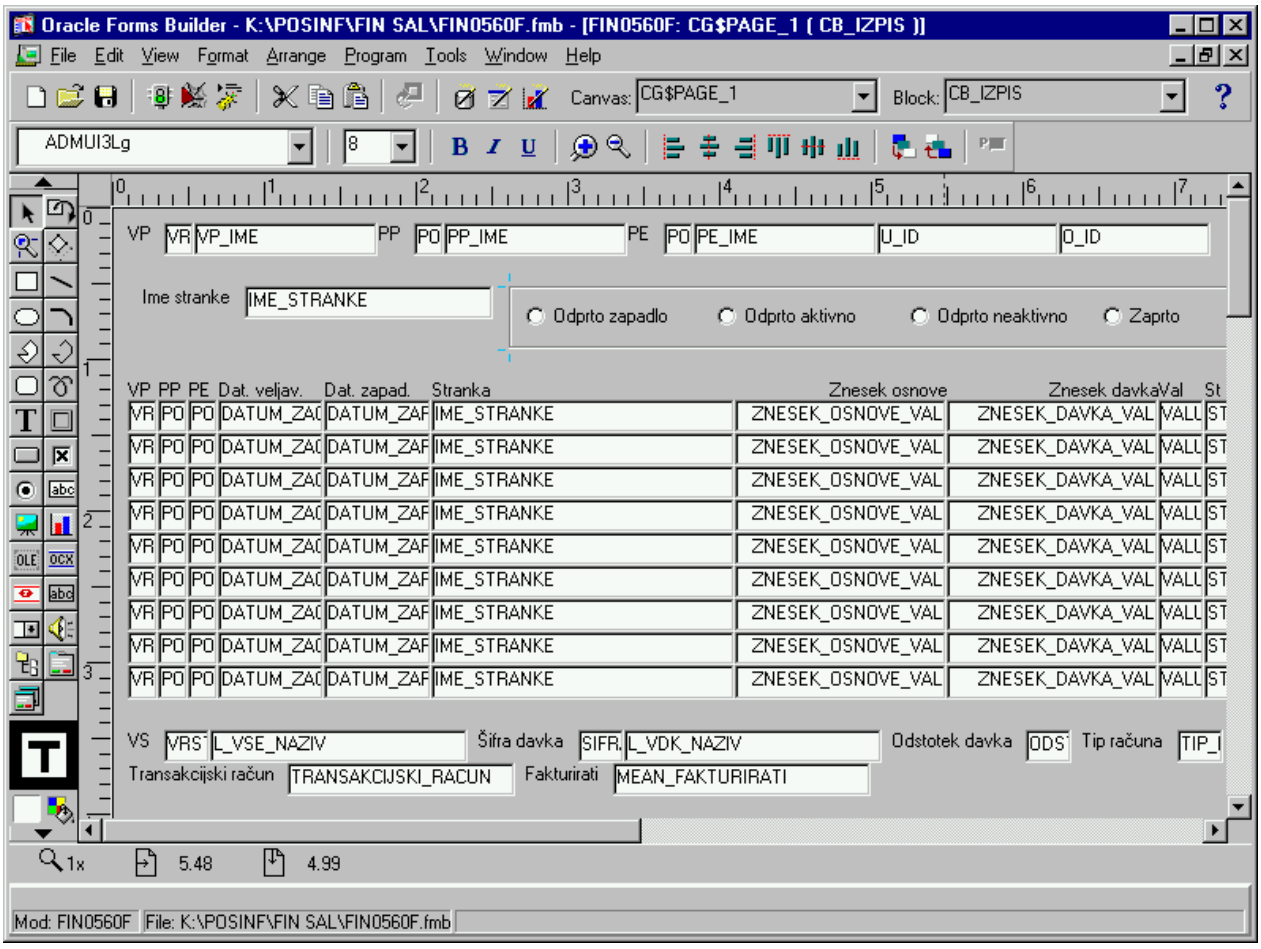
2.1.1.3 PL/SQL urejevalnik

S pomočjo PL/SQL urejevalnika razvijalec piše in preverja/prevaja pravilnost sintakse programske logike. Le-ta lahko vključuje prožilce, funkcije, procedure, ukaze menuja in programske pakete (glej sliko 1, na str. 6).

2.1.1.4 Urejevalnik izgleda

Urejevalnik izgleda je grafični pripomoček za prikazovanje in oblikovanje izgleda uporabniškega vmesnika zaslonskega obrazca. S pomočjo orodne vrstice, ki jo najdemo na levi strani urejevalnika (glej sliko 3, na str. 8), kreiramo in razporejamo objekte po platnu. Izbrane objekte lahko po platnu premikamo z miško, s pomočjo ustreznih tipk ali z določanjem pozicije objektov v lastnostih postavke. Na ta način oblikujemo grafični vmesnik dokler nismo z njim zadovoljni.

Slika 3: Urejevalnik izgleda



Vir: Oracle Form Builder, 2002.

2.1.1.5 Moduli forme

Oracle Forms aplikacijo lahko sestavlja več tipov modulov, kar pomeni, da je aplikacija lahko sestavljena iz samega zaslonskega obrazca/forme, menuja, PL/SQL knjižnic(e) in drugih modulov. Gre za samostojno razvite objekte, ki so medsebojno ločeni in shranjeni v samostojnih datotekah (izvorna datoteka forme je tipa *.fmb, menuja *.mmb, PL/SQL knjižnice pa tipa *.pll). Med razvojem zaslonskega obrazca moramo navesti, kateri menu in knjižnice naj uporablja v delovnem okolju. Če v formi ne definiramo menuja nam Form Builder dodeli privzetega, ki pokriva osnovne funkcionalnosti. Omenjene module lahko vidimo na najvišjem nivoju objektnega navigatorja (glej sliko 1, str. 6). Oracle Forms aplikacija lahko vključuje tudi module iz drugih Oracle Developer (npr. izpisi) in ostalih okolij.
Modularen pristop k razvoju omogoča večjo fleksibilnost pri razvoju novih modulov, pri njihovem vzdrževanju, nadgradnji in izboljšavah.

Zaslonski obrazci/forme
Forme so zbirke objektov in programskih vrstic, ki vključujejo okna, platna, različne postavke (npr. gumbe, pritrdilne postavke), prožilce, procedure in druge objekte. Vsebuje lahko poljubno število objektov in programskih vrstic.
Predstavlja grafični vmesnik preko katerega uporabnik komunicira s podatkovno bazo.

Menu
Predstavlja zbirko menu objektov kot so: glavni menu, padajoči menuji, postavke menuja in orodna vrstica. Uporabniku aplikacije omogočajo enostavnejše delo z zaslonskimi obrazci in drugimi tipi aplikacij.

PL/SQL knjižnice
PL/SQL knjižnice so zbirke procedur, funkcij in programskih paketov. Predstavljajo samostojne module z izvorno datoteko tipa *.pll. Njihov namen je hranjenje programske logike in omogočanje souporabe programske logike večjemu številu aplikacij. Z razliko od programske logike vsebovane v formi, se ta naloži v pomnilnik šele takrat, ko aplikacija uporablja programsko logiko, ki je v njej shranjena. Na ta način izboljšamo delovanje in zmožnosti razvite aplikacije (Oracle Developer 6i, 2002). PL/SQL knjižnice so podrobneje opisane v točki 2.2.1.5 na strani 19.

Objektne knjižnice
So samostojni moduli (izvorna datoteke je tipa *.olb), v katere lahko shranimo objekte namenjene ponovni uporabi. Tako shranjene objekte je mogoče kopirati, z možnostjo dedovanja, s preprosto tehniko povleci in spusti (ang. drag & drop). S tem, ko je določen objekt kopiran z dedovanjem ohrani povezavo z izvornim objektom, kar pomeni, da se bodo spremembe na izvornem objektu poznale tudi v ciljnem objektu.

Pri aktiviranju Oracle Form Builder-ja se objektne knjižnice naložijo same, kar razvijalcu prihrani čas, ki bi ga potreboval, da bi jih naložil v orodje iz datotečnega sistema. Same se naložijo le tiste objektne knjižnice, ki se nahajajo v direktoriju, ki je naveden v registrih operacijskega sistema. Namenjene so standardizaciji in enostavnejšemu kreiranju aplikacij. Podrobnejši opis objektnih knjižnic se nahaja v točki 2.2.1.4 na strani 19.

2.1.1.6 Objekti in nekatere značilnosti Oracle Form Builder-ja

Vgrajeni paketi

Z namenom, da bi bilo programiranje hitrejše in preprostejše, so v Oracle Form Builder vgradili več kot 150 vnaprej pripravljenih procedur in funkcij, ki opravljajo različne standardne funkcionalnosti aplikacij, med drugimi navigacijo med različnimi objekti, proces shranjevanja podatkov, možnost programskega nastavljanja lastnosti objektov, programsko pridobivanje informacij o trenutnih lastnostih objektov in druge.
Poleg obstoječih vgrajenih paketov, lahko sami kreiramo lastne procedure, funkcije ali programske pakete. Ti objekti, bolje znani kot programske enote, so lahko definirani v formah, menujih ali PL/SQL knjižnicah.

Bazni objekti

Veja za bazne objekte v drevesni strukturi objektnega navigatorja omogoča vpogled nad objekti baze podatkov na katero smo trenutno povezani (za razvoj aplikacije je običajno in smiselno, da smo povezani na bazo, kar omogoča lažje kreiranje objektov s pomočjo čarovnikov, sprotno preverjanje pravilnosti sintakse in pravilnosti uporabe baznih objektov). Poleg vpogleda je možno prenašati programsko logiko (procedure, funkcije, pakete) med aplikacijo in bazo s preprosto tehniko povleci in spusti (ang. Drag & Drop).

Podatkovni bloki in postavke

Za izdelavo grafičnega vmesnika sta poleg oken in platen potrebni dve vrsti objektov, in sicer podatkovni bloki in postavke. Postavke so objekti uporabniškega vmesnika, ki uporabniku prikazujejo podatke oz. informacije in mu omogočajo manipulacijo z njimi. Oracle Form Builder ponuja pestro izbiro objektov za oblikovanje uporabniškega vmesnika, med njimi postavke besedila, gumbe, potrditvena polja, postavke seznama, slike, skupine izbirnih gumbov, OLE vsebnike, postavke diagrama, VBX kontrole in druge. S tem razvijalcu omogočajo, da na vizualen in preprost način izdela uporabniku prijazen grafični vmesnik.
Vsaka postavka pripada določenemu bloku, le-ta pa predstavlja nekakšen vsebnik (zbiralnik, zaboj) za postavke. Tudi sam podatkovni blok predstavlja objekt z naborom lastnosti. Prav te lastnosti določajo način kako končni uporabnik komunicira z objekti uporabniškega vmesnika, ki jih ta podatkovni blok vsebuje. Podatkovni blok lahko temelji na bazni tabeli, poizvedbi, proceduri ali vpogledu (ang. view). To pomeni, da se postavke podatkovnega bloka lahko navezujejo na točno določene stolpce v bazi podatkov. Takšna povezava omogoča uporabniku poizvedovanje, vstavljanje, spreminjanje in brisanje zapisov v pripadajoči tabeli brez pisanja ene same vrstice programske logike, če seveda tega izrecno ne

onemogočimo v lastnostih podatkovnega bloka oz. postavk. Bloku, ki ne temelji na objektu podatkovne baze, pravimo nebazni ali kontrolni blok. Podatkovni bloki dejansko predstavljajo nekakšno logično grupiranje postavk, pripadajoče postavke pa so lahko na platnu poljubno razporejene in prikazane v različnih oknih. Forma lahko vsebuje poljubno število blokov, ta pa poljubno število postavk.

Platna
Platno predstavlja podlago zaslonskega obrazca, na katero dajemo objekte. Omogoča dostop do grafične predstavitve zaslonskega obrazca.

Okna
Okna predstavljajo okvir za platna. Vsaka forma ima lahko več oken, v vsakem oknu pa imamo lahko samostojno zaslonsko sliko.

Pripete knjižnice
Ta veja objektnega navigatorja omogoča (so)uporabo programske logike shranjene v PL/SQL knjižnicah. Pripete knjižnice so samostojni programski moduli, ki predstavljajo zbirko programskih enot (procedur, funkcij in programskih paketov).

Programske enote
Predstavljajo objekte za shranjevanje programske logike (procedur, funkcij, programskih paketov). V nasprotju s PL/SQL knjižnicami jih lahko uporabljamo le znotraj posameznega modula.

Opozorila
Opozorilo je modalno okno za prikazovanje sporočil, ki uporabnika seznanjajo z dogajanjem v aplikaciji. S pomočjo sporočil uporabnika obveščamo o nenavadnih situacijah (npr. nepravilen format vnešenega datuma) ali o situacijah, ki bi lahko imele nezaželjene oz. nepričakovane posledice. Obstajajo tri vrste sporočil (Oracle Developer 6i, 2002):

- prekinitev izvajanja programa,
- svarilo,
- nota.

Vsaka vrsta sporočila predstavlja različen nivo pomembnosti opozorila. Pomembnost je razvidna iz edinstvenega znaka (ikone) na modalnem oknu. Tako ima sporočilo, ki nas opozori o prekinitvi izvajanja programa, za edinstveni znak klicaj.

Lista vrednosti
Predstavlja okno z naborom vrednosti, ki temelji na poizvedbi ali statični zalogi vrednosti. Uporablja se kot pomoč in kontrola uporabniku pri vnašanju podatkov. Načeloma je pripeta na postavko v katero želimo vnašati podatek, lahko pa jo prikazujemo tudi programsko.

Skupina zapisov
Predstavlja objekt, ki je tesno povezan z listo vrednosti. V njem je definirana poizvedba (SQL stavek), na podlagi katere se v listi vrednosti prikazujejo izbrani podatki. Vrednosti so lahko določene tudi statično s fiksnim naborom vrednosti (npr. DA, NE).

Skupina objektov
Ta veja objektnega navigatorja predstavlja nekakšen »zbiralnik« za skupine objektov. Definiramo jo, ko želimo združiti med seboj povezane objekte, z namenom navadnega kopiranja ali kopiranja z dedovanjem v druge module.

Vizualne lastnosti
Gre za objekt v katerem so definirane in shranjene vizualne lastnosti objektov kot so vrsta, oblika, velikost pisave, barva objektov in druge lastnosti. To nam predvsem koristi pri kreiranju novih objektov. Namesto, da določamo vrednosti posameznim lastnostim na novo kreiranega objekta, mu preprosto povemo naj gre iskat nastavitve/lastnosti v objekt tipa vizualne lastnosti. S tem prihranimo veliko časa in omogočamo standardizacijo izgleda.

Razredi lastnosti
Namen tega objekta je podoben prejšnjemu, le da tukaj ne gre samo za vizualne lastnosti, temveč za vse lastnosti objekta. Predstavlja objekt s skupino lastnosti in njihovimi vrednostmi. Ko ga enkrat kreiramo lahko druge objekte baziramo nanj, tako da nov objekt kopira ali deduje lastnosti definirane v njem.
Kot samostojne objekte jih lahko kopiramo med moduli, prav tako pa jih lahko kopiramo z namenom dedovanja v poljubno število modulov.

Parametri
Parametri predstavljajo objekte na nivoju forme v katere shranjujemo vrednosti različnih tipov podatkov (npr. število, datum). Te vrednosti so lahko določene kot izhodiščne, začetne vrednosti, lahko pa jih določamo tudi dinamično med izvajanjem programske logike. Lahko jih uporabimo kot substitut za globalne spremenljivke, v primerjavi z njimi pa manj obremenjujejo pomnilnik. Na parametre se lahko sklicujemo iz vseh nivojev forme. Pogosto jih uporabljamo takrat, ko pričakujemo, da bi se lahko določen podatek v programski logiki pogosteje spreminjal (npr. stopnja DDV). V takem primeru definiramo nov parameter in se v programski logiki sklicujemo nanj. V primeru spremembe podatka (npr. stopnja DDV se zviša iz 20 na 23 odstotkov), ki ga vsebuje parameter je dovolj, da ga spremenimo v parametru in ne v programski logiki. Na ta način prihranimo veliko truda in zmanjšamo možnost napak.

Dogodkovno vodeno programiranje
Oracle Form Builder podpira model dogodkovno vodenega programiranja. Enkrat, ko smo definirali osnovno strukturo in funkcionalnosti aplikacije s kreiranjem objektov in nastavitvijo njihovih lastnosti, lahko njeno funkcionalnost nadgradimo s pisanjem programske logike. Programiranje v Form Builder-ju poteka v programskem jeziku PL/SQL (ang. Oracle's

procedural languagae extantion to SQL, the relational database language), to je v Oraclovemu proceduralnemu jeziku, ki predstavlja podaljšek jezika SQL (ang. Structured Query Language), jezika relacijskih baz. Programski jezik PL/SQL združuje zmožnosti SQL-ovega manipuliranja (poizvedovanje, shranjevanje, brisanje, spreminjanje podatkov) in transakcijskega procesiranja s konstrukti, ki so tipični za jezike proceduralnega programiranja. Ti zajemajo deklariranje spremenljivk, konstant, prirejanje vrednosti, zanke, pogojno izvajanje in drugo.
Oracle Form Builder vključuje integrirano in interaktivno »razhroščevalno okolje« (ang. debugging environment), ki omogoča spremljanje, zaustavitev izvrševanja programa, pregled programskih vrstic in druge uporabne funkcionalnosti.

Prožilci

Prožilec je programska enota, ki se izvrši ob točno določenem dogodku. Pripeti so lahko na različne objekte (forma, blok ali postavka) in na različnih nivojih. Ti so trije: nivo forme, nivo bloka in nivo postavke. Najvišji nivo predstavlja nivo forme, najnižjega pa nivo postavke. Vsak prožilec, ki ga definiramo, je povezan s točno določenim dogodkom. Form Builder ponuja široko paleto dogodkov za katere se lahko sprožijo prožilci. Grupirani so v šest večjih skupin:

- dogodki povezani s poizvedbo,
- dogodki povezani z vnosom podatkov in njihovim preverjanjem,
- dogodki povezani z navigacijo in miškinimi premiki,
- dogodki povezani z interakcijo s postavkami forme,
- dogodki povezani z notranjimi dogodki v formi,
- dogodki v zvezi z napakami in sporočili.

Nanje odgovarjamo z naslednjimi skupinami prožilcev (Oracle Developer 6i, 2002):

Query-time tirggers	Sprožijo se tik pred in tik zatem, ko uporabnik oz. aplikacija izvrši poizvedbo na bloku. Primer: Pre-Query, Post-Query.
Validation triggers	Ta tip prožilcev se sproži, ko aplikacija preverja podatek postavke oz. zapisa. Preverjanje se izvaja med navigacijo, ki je lahko posledica vnosa uporabnika, programske kontrole ali privzetega procesiranja. Primer: When–Validate-Item, When–Validate-Record.
Navigational triggers	Sprožijo se kot odgovor na dogodke povezane z navigacijo, ko se vnosni fokus/kurzor premakne iz trenutnega objekta na ciljni objekt. Navigacijski dogodki se pojavljajo na različnih nivojih: nivo forme, bloka, zapisa ali postavke. Primer: Pre-Form.
Master-detail triggers	Ta tip prožilcev kreira Oracle Form Builder, ko definiramo povazvo med dvema blokoma, lahko pa jih izdelamo tudi sami. Omogočajo koordinacijo med zapisi dveh blokov. Primer: On-Check-Delete-Master, On-Populate-Details

Message-handling triggers	Sprožijo se kot odgovor na dogodke povezane s sporočili. Oracle Form Builder samodejno kreira sporočila o napakah in informacijah kot odgovor na dogodke ob izvajanju programa. Primer: On-Error, On-Message.
Transactional triggers	Sprožijo se, ko aplikacija dela s podatki. Primer: On-Delete, On-Insert.
When-New-Instance-triggers	Ta vrsta prožilcev se sproži ob koncu navigacijske sekvence, ki postavi fokus na drugo postavko, zapis, blok ali formo. Primer: When-New-Item-Instance, When-New-Record-Instance.
Interface event triggers	Sprožijo se ob dogodkih povezanih z vmesnikom forme. Nekateri med njimi, kot npr. When-Button-Pressed se sprožijo samo kot odgovor na uporabnikovo interakcijo (vnos), drugi, npr. When-Window-Activated, pa se lahko sprožijo ob vnosu uporabnika ali programske kontrole.
Block processing triggers	Sprožijo se ob dogodkih povezanih z urejanjem zapisov bloka. Primer: When-Clear-Block, When-Remove-Record

Primer dogodka: pritisk na gumb. Pri tem se sproži prožilec npr. When-Button-Pressed, katerega pripnemo na objekt tipa gumb. Prožilec, in s tem programska logika, ki je v ozadju, se sproži ob vsakem pritisku na objekt gumb.

Delitev aplikacije

Proceduralni jezik PL/SQL se uporablja tako na strani odjemalca (Oracle Forms in druge aplikacije) kot na strani strežnika (bazni prožilci, v bazi shranjene procedure, funkcije, podatkovni paketi). S tem je razvijalcu omogočeno, da lahko izvajanje programske logike porazdeli med odjemalcem in strežnikom, kar ima lahko velik vpliv na boljše delovanje aplikacije. S porazdelitvijo zmanjšamo promet med odjemalcem in strežnikom, ker se del programske logike izvršuje na strani strežnika, zmanjšamo pa tudi obremenjenost pomnilnika odjemalca. Običajno je strežnik zmogljivejši računalnik (v primerjavi z odjemalcem), kar dodatno pozitivno vpliva na boljše odzivne čase aplikacije, ki ima na tak način porazdeljeno programsko logiko.

Prenos programske logike med odjemalcem in strežnikom je v objektnem navigatorju omogočen tudi s tehniko potegni in spusti (ang. Drag & Drop).

2.2 Oracle Designer

Oracle Designer je enotno ime za skupino programskih orodij, namenjenih analizi poslovnih potreb ter oblikovanju in generiranju podatkovne zbirke in programskih rešitev za okolja odjemalec/strežnik oz. svetovni splet. V sistemski analizi nudi podporo modeliranju poslovnih procesov, entitet in relacij, poslovnih funkcij, podatkovnih tokov, BPR (ang. Business Process Reengineering), v sistemskem oblikovanju pa omogoča modeliranje podatkovne zbirke in programskih modulov (Žabkar, 2000, str. 35).

Predstavlja novo generacijo orodij I-CASE (ang. Integrated Computer-Aided Software Engineering), temelječih na ideji repozitorija, s pomočjo katerega sistemski analitiki in razvijalci shranjujejo in zajemajo poslovne potrebe, pravila, oblikujejo podatkovne strukture, ki ustrezajo tem poslovnim potrebam ter jih prenašajo v podatkovne zbirke.

Definicija repozitorija: repozitorij je globalna shramba oz. zbiralnik za hranjenje in organiziranje vseh informacij o informacijskem sistemu.

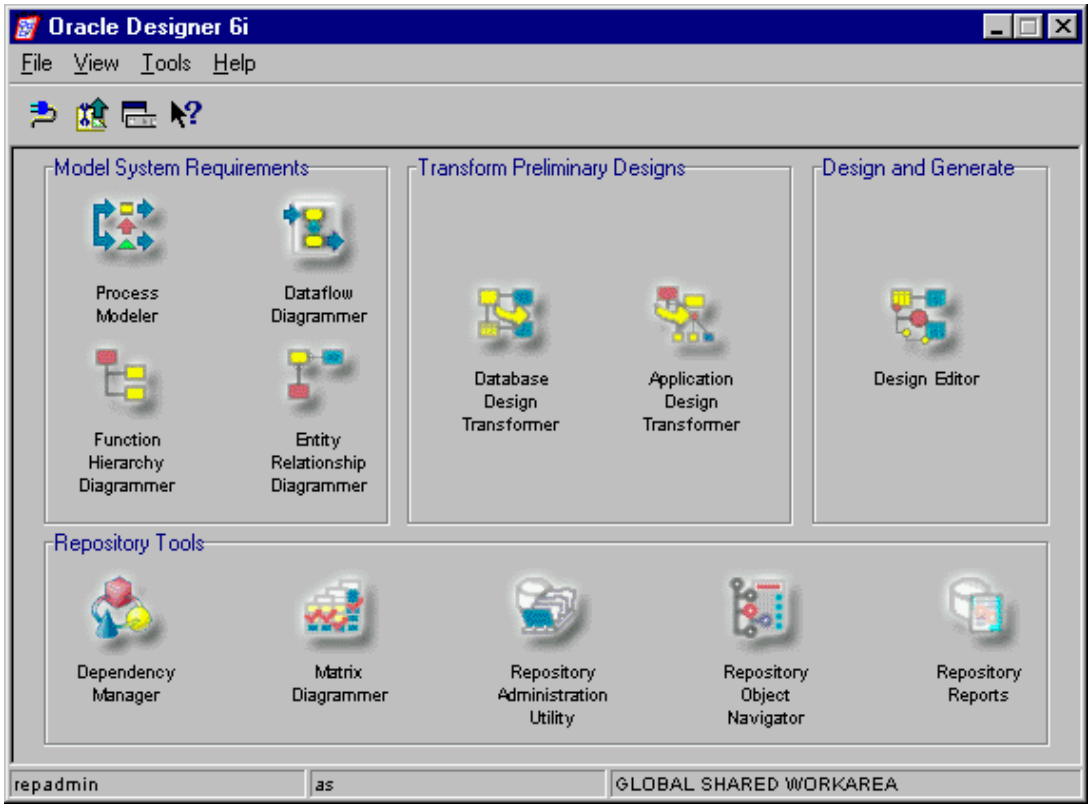
Repozitorij predstavlja večuporabniško okolje, ki je zgrajeno na principu standardne Oraclove baze, katera podpira arhitekturo odjemalec-strežnik, kar pomeni, da se repozitorij lahko nahaja na oddaljenem strežniku, le ta pa lahko uporablja različne operacijske sisteme, kot so Windows 95/98/NT/2000, Sun Solaris, IBM R6000 AIX, HP UX, Irix SCO Unix, Netware, Linux in mnoge druge. Vsa orodja/pripomočki Oracle Designer-ja so grafična, med njimi tudi repozitorijev objektni navigator, prek katerega je uporabnikom omogočeno enostavno manipuliranje s podatki v njem.

Repozitorij omogoča ponovno uporabo njegovih elementov, kar je izrednega pomena za konsistentnost, vzdrževanje kot tudi prispevek k večji produktivnosti in hkrati predstavlja vir informacij za sistemske analitike, projektne vodje in razvijalce programskih rešitev.

Orodja Oracle Designer-ja so združena v vsebinske sklope, ki odsevajo potrebe različnih uporabnikov: ravnateljev, sistemskih analitikov, projektnih vodij in razvijalcev programskih rešitev. Struktura Oracle Designer-ja temelji na odprtem repozitoriju, ki dopolnjen z orodji za modeliranje in generiranje sistemov, predstavlja izvor sinergičnih učinkov vseh prej naštetih uporabnikov. V okviru programske zbirke je podprt celoten razvojni proces informacijske podpore. Z uporabo različnih orodij, prilagojenih posameznim fazam razvoja računalniške podpore lahko razvijalci in oblikovalci informacijskega sistema hitro in učinkovito razvijajo in spreminjajo definicije informacijskega sistema. Programsko zbirko sestavljajo naslednji sklopi orodij:

- orodja sistemske analize,
- orodja za pretvorbo prvotnih načrtovanj,
- orodja za razvoj in programski generatorji,
- orodja repozitorija.

Slika 4: Struktura razvojnega okolja Oracle Designer



Vir: Oracle Designer, 2002.

V splošnem se orodja CASE od predhodnih orodij razlikujejo v naslednjih pomembnih vidikih (Heričko, 1999, str. 10):

- podpora namenskemu, osebnemu računalniškemu okolju,
- uporaba grafike pri specifikaciji in dokumentiranju informacijskih sistemov,
- s pomočjo računalnika zajemanju in povezovanju vseh informacij o razvijajočem informacijskemu sistemu od začetnih zahtev pa tja do aktivnosti vzdrževanja,
- uporaba neke vrste umetne inteligence za izvajanje oz. avtomatizirano vršenje/izvajanje mnogih rutinskih opravil pri razvoju in vzdrževanju.

Pričakovanja glede tehnologije CASE – imenovane tudi tehnologije avtomatizacije so bila in so še vedno precejšnja. Pridobitve, ki jih CASE nudi razvijalcem programskih sistemov pa so naslednja (Heričko, 2000, str. 1):

- praktičnost strukturnih in objektnih tehnik,
- vsiljuje programsko/informacijsko inženirstvo,
- izboljšuje kakovost programske opreme (avtomatske kontrole, preverjanja),
- praktičnost prototipiranja,
- lažje, enostavnejše vzdrževanje,
- skrajšan čas razvoja,
- razvijalci se lahko osredotočijo na kreativni del razvoja,

- ponovna uporaba programskih komponent,
- podpora pri izdelavi sistemske dokumentacije

Čeprav ima Oracle Designer več različnih generatorjev (strežnikov generator, generator form, generator izpisov, generator za Visual Basic, generator C++ razredov ter generator spletnih aplikacij), se v skladu z razlago v uvodu diplomskega dela v nadaljevanju osredotočam na generator form.

2.2.1 Generator form

Pri pregledovanju literature sem večkrat naletel na dilemo pri uvrščanju CASE Oracle Designer-ja v RAD okolje oz. med generatorje programskih rešitev.
Pri generatorjih programskih rešitev je namreč potrebno natančno opredeliti, kaj pod tem razumemo. Razhajanje mnenj izhaja iz različnega pojmovanja generatorjev v strokovni literaturi. Osebno sem generator programskih rešitev opredelil kot del orodja CASE Oracle Designer, ker je generiranje pri Oracle Designer-ju vodeno preko definicij modulov v repozitoriju in vključuje PL/SQL knjižnice, preference, objektne knjižnice in forms predloge. V ožjem smislu je tudi Oracle Form Builder generator, saj interaktivno delo z bazo podatkov preoblikuje v generirane SQL stavke (vnos, poizvedovanje, brisanje, ažuriranje), zato nekateri avtorji uvrščajo generatorje programskih rešitev med orodja RAD. Po mojem mnenju je pomemben kriterij za to, kam uvrstiti generator programskih rešitev, le vodljivost postopka generiranja. Le-ta lahko temelji na vodljivosti preko repozitorija (takrat je generator del orodja CASE) ali pa s tem razumemo, da generiranje kode poteka s pomočjo čarovnikov (ang. wizard) brez vpetosti v repozitorij in brez povezanega okolja za analizo in oblikovanje informacijskega sistema. Takrat je generator del orodja RAD. Orodja RAD ne omogočajo oblikovanje baze podatkov in predpostavljajo, da je le-ta že narejena. Orodja CASE pa pokrivajo celoten življenjski cikel razvoja programske rešitve (strategija, analiza, oblikovanje, dokumentacija, gradnja prototipa, uvajanje, produkcija).

Definicija generatorja programov: celota računalniških programov v jeziku četrte generacije, ki omogoča, da računalnik samodejno izdela računalniški program na podlagi zahtev uporabnika (Ivan Turk et al., 1987, str. 66).

Za pravilno generiranje zaslonskih obrazcev v Oracle Designer-ju se je potrebno držati predpisanega postopka, ki vključuje večje število korakov in informacij o tem, kaj in kako bomo generirali. Ker predstavlja ta postopek zahteven proces, v katerem sodeluje večje število dejavnikov oz. komponent, je smiselno natančneje opredeliti vsakega od teh dejavnikov in pojasniti njihovo vlogo ter vpliv na proces generiranja. Pri procesu generiranja sodelujejo sledeči dejavniki oz. komponente Oracle Designer-ja:

1. repozitorij Oracle Designer-ja,
2. generatorjeve prednostne nastavitve (GPN),
3. Oracle Developer šablona,

4. objektne knjižnice,
5. PL\SQL knjižnice,
6. urejevalnik oblikovanja,
7. Headstart Oracle Designer.

2.2.1.1 Repozitorij

Repozitorij Oracle Designer-ja predstavlja centralno skladišče definicij oz. zbirko opisnih informacij (meta podatki – podatki o podatkih) o vseh vrstah podatkov, ki so potrebne pri gradnji informacijskega sistema. V njem je shranjena definicija bodočega modula, ki ga želimo generirati. Le-ta pove generatorju form kaj naj generira (št. blokov, postavk, oken, platen,...). Posamezni gradniki aplikacije opredeljujejo ustroj in funkcionalnost bodoče generirane programske rešitve. Kakovosten repozitorij mora uporabnikom nuditi naslednje funkcionalnosti:

- zanesljivost in stopnjevana prilagodljivost,
- odprtost in razširljivost,
- enostavnost upravljanja,
- prilagodljiva podpora skupinskemu delu,
- bogat podatkovni meta model,
- tesna povezanost z razvojnimi orodji.

2.2.1.2 Generatorjeve prednostne nastavitve

GPN - generatorjeve prednostne nastavitve so parametri, ki krmilijo generatorje programskih rešitev pri oblikovanju uporabniškega vmesnika in vplivajo na nekatere standardne funkcionalnosti programskih rešitev, kot so javljanje napak, upravljanje z okni, navigacija, prenos parametrov med programskimi moduli, idr. S pomočjo prednostnih nastavitev generatorju povemo, kako naj generira v repozitoriju shranjene definicije programskih rešitev. Prednostne nastavitve so organizirane v drevesno strukturo na štirih nivojih, in sicer na nivoju:

- aplikacije,
- domene, tabele in modula,
- omejitev, stolpcev in komponent modula,
- uporabe omejitev, polj in skupnih polj.

2.2.1.3 Oracle Developer šablona

Predstavlja ogrodje bodoče generirane forme. Gre za standardni del bodočih zaslonskih obrazcev, ki vsebuje splošne definicije grafičnih objektov in postopkovne PL\SQL logike, ki se bodo pojavile v večini bodočih generiranih zaslonskih obrazcev. Prav zaradi tega, ker se bo ta del večkrat ponovil, ga je smiselno vnaprej narediti, kajti šablona bo predstavljala izhodišče za gradnjo novih zaslonskih obrazcev. Tako razvijalcem ni potrebno znova in znova razvijati

standardnih delov aplikacije, temveč na že narejeno šablono dodajajo definicije in funkcionalnosti novega modula. Poleg tega, da poenostavlja delo razvijalca, prinaša standardno podobo nekaterih delov njenega uporabniškega vmesnika, kot so orodna vrstica, prikazovanje sporočil uporabniku, koda za prikazovanje tipičnih napak, koda za podporo delu z miško, idr. Ime Developer šablona izhaja iz tega, ker je šablona narejena in se uporablja v okolju Oracle Developer, prav tako pa predstavlja nepogrešljivo komponento za generator form, kajti v procesu generiranja postane šablona osnova, na katero generator dodaja definicije iz repozitorija na podlagi prednostnih nastavitev. Šablona ni shranjena v repozitoriju, temveč se nahaja na odjemalčevi strani, nameščena na datotečni sistem. Če želimo šablono kakorkoli spremeniti (nekaj jih namreč dobimo z orodjem Oracle Designer) oz. narediti čisto novo, je to mogoče le v okolju Oracle Developer. Šablona predstavlja temelje bodoče aplikacije, zato je dobra priprava šablon eden od ključnih dejavnikov, ki vplivajo na uspešnost generiranja form. Brez šablone Oracle Designer ne more generirati form.

2.2.1.4 Objektne knjižnice

Objektna knjižnica je zbirka objektov za katere predvidevamo, da se bodo lahko pojavili na nekaterih ali celo vseh generiranih formah. Zato jih kot vnaprej pripravljene s pomočjo kopiranja oz. dedovanja integriramo v bodoče module. Slednja lastnost (dedovanje) pa se od kopiranja razlikuje v tem, da ohrani povezavo z izvornim objektom v objektni knjižnici. Če izvirni objekt spremenimo se bo to posledično poznalo tudi na vseh objektih, kateri so dedovali lastnosti tega objekta, vendar pod pogojem, da generirano formo ponovno zgeneriramo oz. zadostuje, da jo ponovno prevedemo v okolju Oracle Developer. Med prevajanjem dedovani objekt prevzame nove lastnosti objekta v knjižnici. Dedovanje je zato primerno za objekte, ki se pogosto spreminjajo, hkrati pa se pojavljajo v več modulih. S pomočjo objektne knjižnice in dedovanja tako dosežemo večjo standardizacijo, ponovno uporabo standardov in objektov v večjem številu form in cenejše vzdrževanje sistema.

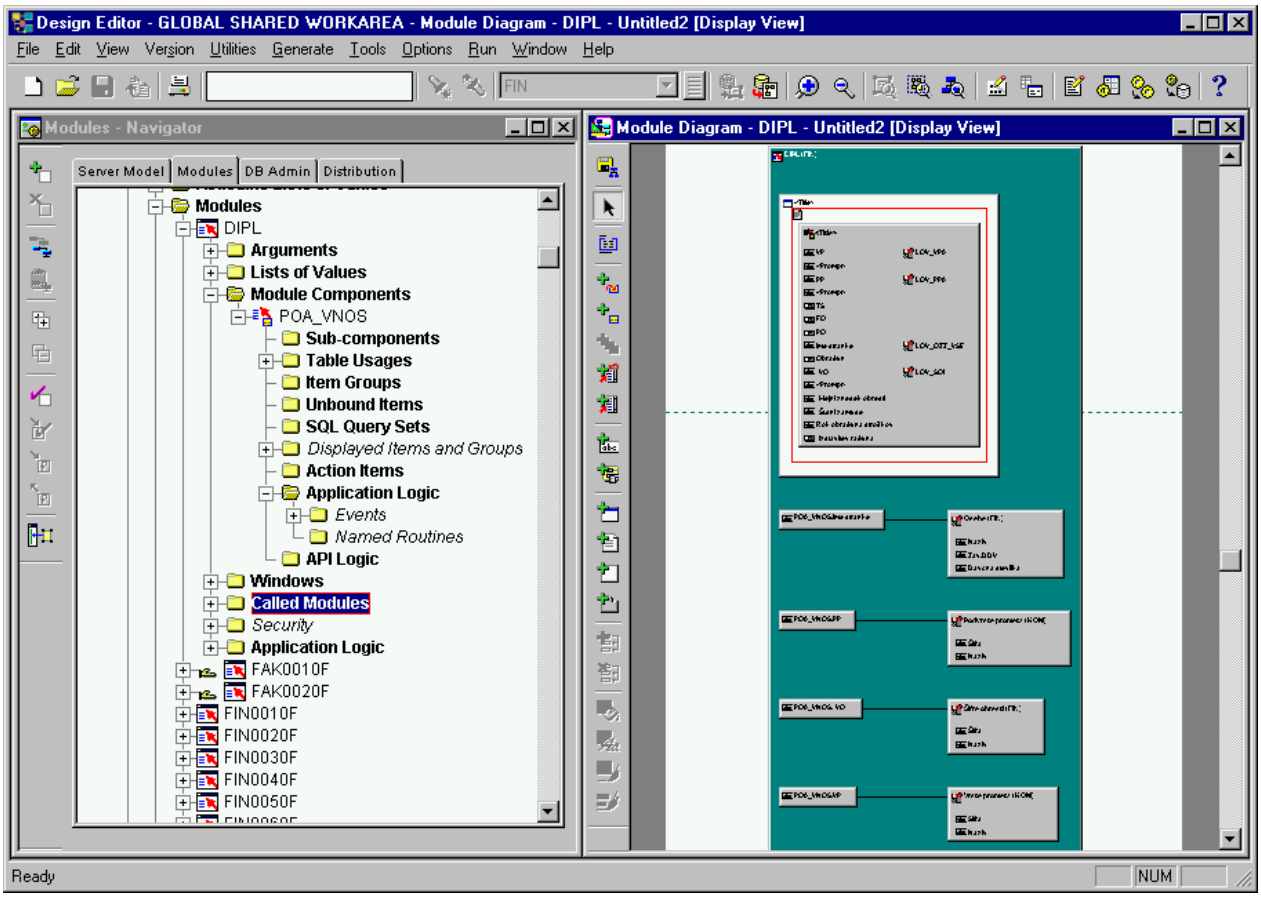
2.2.1.5 PL\SQL knjižnice

Predstavljajo module v katerih shranjujemo PL\SQL programsko logiko, čeprav jo lahko v zaslonskih obrazcih pišemo in uporabljamo brez uporabe knjižnic. Prednost PL\SQL knjižnic je ta, da lahko programsko logiko, ki je shranjena v njej kličemo iz več aplikacij, kar zmanjšuje podvajanje in spodbuja k ponovni uporabi. PL\SQL knjižnica sodeluje pri generiranju bodoče programske rešitve s tem, da se pri prevajanju, ki poteka ob generiranju forme, preverja ali se programska logika morda nahaja v knjižnici, ne pa tudi pravilnost programske logike. Shranjene so na datotečnem sistemu in ne v repozitoriju.

2.2.1.6 Urejevalnik oblikovanja

Gre za tisti del orodja Oracle Designer-ja skozi katerega lahko vidimo in manipuliramo vsebino celotnega repozitorija. S pomočjo štirih zavihkov lahko izbiramo med različnimi tematskimi sklopi oblikovanja podatkovne zbirke in programskih rešitev. Za samo izdelavo zaslonskih obrazcev je namenjen zavihek za module. Sestavljajo ga diagram modulov in grafični del urejevalnika oblikovanja (glej sliko 5, na str. 20), kateri omogoča grafično oblikovanje ter približen prikaz izgleda generiranega modula. Tako kot večina navigatorjev je tudi navigator modulov urejen v drevesno strukturo in omogoča pregledovanje objektov na več nivojih. Če si želimo določen objekt iz navigatorja ogledati v grafičnem delu urejevalnika oblikovanja, ga enostavno z miško povlečemo vanj, grafični urejevalnik pa poskrbi, da se nam objekt odpre v primernem grafičnem diagramu. Torej je v urejevalniku oblikovanja objekte možno videti in z njimi manipulirati na dva načina, in sicer skozi diagram modulov ter skozi grafični del urejevalnika oblikovanja.

Slika 5: Urejevalnik oblikovanja



Vir: Oracle Designer, 2002.

2.2.1.7 Headstart Oracle Designer

Headstart Oracle Designer ni standardni del Oracle Designer-ja, temveč predstavlja dodatek in nadgradnjo s pomočjo katerega lahko bistveno skrajšamo čas, ki je potreben za izdelavo in

uvajanje prototipne programske rešitve. S Headstartom namreč dobimo že narejene prototipe (šablone, objektne knjižnice, PL/SQL knjižnice, idr.), od nas pa je odvisno ali smo s ponujenimi prototipi zadovoljni oz. če ustrezajo načrtovanim potrebam programske rešitve. V primeru, da nam prototipi odgovarjajo se lahko takoj osredotočimo na poslovne potrebe programske rešitve, sicer jih lahko po potrebi dopolnimo, prilagodimo specifikam programske rešitve, v skrajnem primeru pa naredimo čisto nove, kar pomeni da Headstart Oracle Designer-ja ne potrebujemo. Sestavljen je iz dveh komponent:

- Headstartov paket predlog (ang. Headstart Template Package) predstavlja skupek naprednih predlog in ponovno uporabnih komponent, ki omogočajo 100% generiranje profesionalnih grafičnih vmesnikov za enostavno uporabo v MS Windows, svetovnem spletu in drugih okoljih. Sestavljajo ga:
    - paket Headstartovih šablon,
    - paket Headstartovih ponovno uporabnih komponent,
    - paket Headstartovih zmogljivih PL/SQL knjižnic,
    - paket Headstartovih optimalno prednostno nastavljenih navodil za generatorje form, izpisov, menujev in sprotne pomoči,
    - Headstartov paket izboljšanih objektnih knjižnic,
    - paket pomožnih programov
- Headstartovi pripomočki (ang. Headstart Utilities), garnitura, ki preverja in uveljavlja (vsiljuje - v pozitivnem smislu) Oraclove svetovno razširjene razvijalske standarde in bistveno pospešuje razvoj skozi avtomatizacijo časovno potratnih nalog v Oracle Designer-ju.

2.2.1.8 Koncept popolnega generiranja programskih rešitev

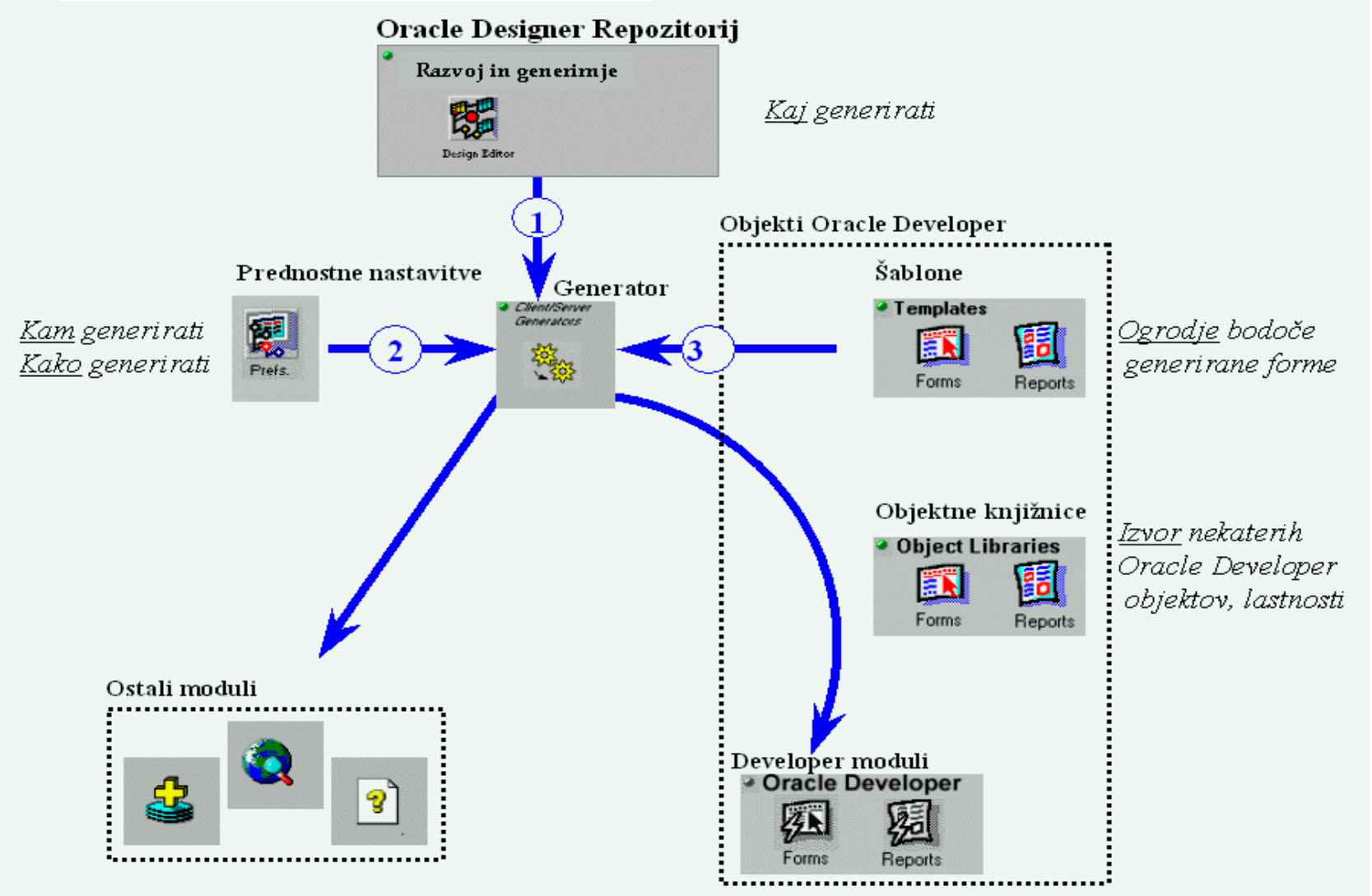
Popolno ali 100% generiranje pomeni, da formo v celoti generiramo preko Oracle Designer-ja brez kakršnihkoli zunanjih posegov v formino podobo ali njeno PL/SQL programsko logiko. Z zunanjim posegom se pojmuje poseg v izvirno kodo genereirane forme preko okolja Oracle Developer. Ta omogoča pregledovanje, analiziranje in popravljanje vseh gradnikov generirane forme. Vsak zunanji poseg na formi ne bo shranjen v repozitoriju, to pa pomeni, da bomo ob ponovnem generiranju vse rezultate zunanjega posega izgubili. Smisel popolnega generiranja je v tem, da lahko modul generiramo iz repozitorija brez potrebe po zunanjih posegih.

Definicija 100% generiranja (Davelaar, B.l., str. 1): Forma je 100% generirana, če jo lahko po spremembi

1. v šabloni,
2. v prednostnih nastavitvah,
3. v definiciji tabele ali stolpca,
4. v definiciji uporabe tabel programskega modula

ponovno generiramo iz repozitorija Oracle Designer-ja, brez zunanjih posegov v izgled ali generirano izvorno kodo.

Slika 6: Potrebne informacije za generiranje forme



Vir: Atkins Ken, 1999.

3 PRIMERJAVA ORODIJ: ORACLE FORM BUILDER IN ORACLE DESIGNER

Orodji za razvoj zaslonskih obrazcev sta med seboj tesno povezani in na določen način odvisni drug od drugega. Povezanost teh dveh orodij zagotavlja organizacijam lažje delo pri uvajanju sprememb iz poslovnega okolja v generirano programsko rešitev. Natančneje povedano je Oracle Designer odvisen od nekaterih komponent (Developer šablona, objektna knjižnica, idr.), ki so narejene v Oracle Developer okolju, kar pomeni, da brez njih Oracle Designer ne more generirati aplikacij.
Te komponente dobimo že narejene v paketu Oracle Headstart Designer, vendar se lahko kaj hitro pojavi potreba po prilagoditvi lastnim potrebam, ki pa jih je mogoče izvesti le v Oracle Developer okolju. Dejansko je Oracle Designer orodje s pomočjo katerega generiramo Oracle Developer forme, zaradi česar je podobnost in odvisnost orodij očitna.

Zaradi omenjene povezanosti bom primerjal tiste dele in gradnike orodij, ki se po mojem mnenju najbolj razlikujejo oz. jih najdemo le pri enem izmed orodij ter tiste dele orodij, ki najbolj vplivajo na večjo produktivnost razvijalcev. Pri tem mislim predvsem na vsebinske razlike, s pomočjo katerih bom skušal izpostaviti prednosti in slabosti posameznega orodja.

V primerjavi bo v ospredju Oracle Designer, ki je kot CASE orodje drugače zasnovano in temu primerno ponuja širši spekter pripomočkov/orodij v primerjavi z Oracle Form Builder-jem. Pri tem moram poudariti, da so prav tiste komponente, ki jih najdemo le pri enem od orodij, največkrat odločilni dejavnik večje produktivnosti razvijalcev.

V nadaljevanju bom primerjavo zožil na izbrana področja oz. komponente orodij:

- Headstart Oracle Designer,
- analiza odvisnosti,
- sistemska dokumentacija,
- repozitorij Oracle Designer-ja,
- ponovna uporaba komponent,
- shranjevanje verzij modulov,
- čarovniki,
- podpora skupinskemu delu,
- razvoj zaslonskih obrazcev in oblikovanje grafičnega vmesnika aplikacije.

Določeni vidiki primerjave, predvsem struktura in delovanje orodij, sta razvidna iz opisov v drugem poglavju (glej 2. poglavje: Predstavitev razvojnih orodij).

3.1 Headstart Oracle Designer

Headstart Oracle Designer lahko temeljito skrajša čas, ki je potreben za izdelavo in uvajanje prototipne programske rešitve, generirane preko Oracle Designer-ja, v produkcijsko okolje. Namen paketa Headstart Oracle Designer je omogočiti uporabnikom, da na enostaven način, brez večjih zapletov, začnejo uporabljati orodje Oracle Designer. Na ta način se razvijalci lažje osredotočijo na poslovne potrebe programske rešitve, namesto da bi večino časa pridobivali le tehnološka znanja o samem orodju. Rezultat tega so časovni in finančni prihranki ter boljša kakovost in lažje vzdrževanje generirane programske rešitve. Razvijalcem nudi oporo in zgled pri razvijanju lastnih rešitev, hkrati pa vpeljuje Oraclove svetovno razširjene razvijalske standarde (Oracle CDM Advantage[3]).

Oracle Headstart Designer nam ponuja popolnoma delujoče in zanesljive komponente. Med njimi je tudi napreden menu z naslednjimi funkcijami in komponentami (Oracle Consulting, 1998, str. 3):

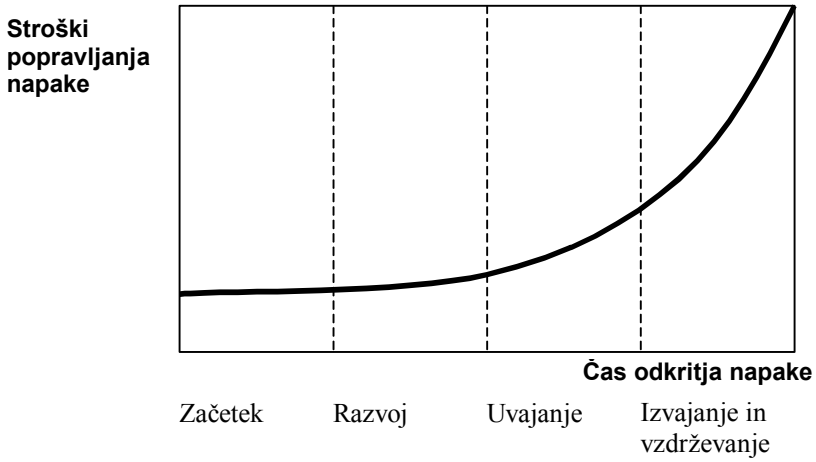
- orodna vrstica, ki jo po potrebi priredimo za lastne potrebe brez ponovnega generiranja iz Oracle Designer-ja,
- menu s standardnimi funkcijami (Datoteka, Urejanje, Okno, Pomoč, idr.),
- dinamično omogočanje in onemogočanje gumbov, postavk menuja,
- generiranje iskalnih oken kot alternativa poizvedbam,
- vizualne indikatorje postavk (obvezno polje, samo za branje, dovoljena poizvedba),
- dinamično imenovanje in pozicijoniranje oken,
- okno s koledarjem kot pomoč uporabniku za lažji vnos datumov,
- možnost spreminjanja barve aplikacije, trenutnega zapisa, jezika, idr.,
- okno za prikaz informacije o trenutnem zapisu,
- forma za klicanje izpisov s parametri,
- napreden večjezični sistem za obvladovanje sporočil, ki razkriva vzrok in potrebno dejanje za odpravo vzroka,
- forma za zamenjavo gesla,
- forma za prikazovanje informacij o trenutni aplikaciji (ime, verzija),
- »razhroščevalno« okno za hitro odkrivanje vzrokov težav,
- drugo.

Po mnenju Oraclovega svetovalnega oddelka, lahko na podlagi Oracle Headstart paketa prihranimo od deset do petnajst tednov razvoja in zagotovimo optimalno uporabo Oracle

[3] Oracle CDM Advantage je metodologija, ki nas vodi skozi celoten življenjski cikel prenove informacijskega sistema in izdelave aplikativne rešitve. Vsebuje definicije procesov, faz, aktivnosti, njihove medsebojne povezave in potrebne resurse ter postopke, ki so potrebni za nadzor kakovosti. Sestavni del metodologije so tudi vzorci izhodnih dokumentov vsake posamezne aktivnosti ter standardi in priporočila, ki jih je potrebno upoštevati pri razvoju aplikativne rešitve. Obstajata dva pristopa, in sicer Classic in Fast track, ki se uporabljata v odvisnosti od velikosti projekta (Madžarević, 2001, str. 21).

Designer-jevih generatorjev (Oracle Consulting, 1998, str. 2). S pomočjo Headstartovih pripomočkov prihranimo dodaten čas, ki bi ga sicer potrebovali za kontrolo in implementacijo razvijalskih standardov, s tem pa smo, poleg prihranitve časa, povečali tudi kvaliteto informacijskega sistema. Na ta način postane tveganje kvalitete končnega proizvoda mnogo manjše.
Spodnji graf prikazuje stroške popravljanja napak glede na čas njihovega odkritja, s pomočjo Oracle Headstart paketa pa naredimo prve korake k preprečevanju takšnih napak že v samem začetku. To seveda ne velja za vsebinske napake.

Slika 7: Stroški popravljanja začetnih napak glede na čas odkritja



Vir: Gradišar, Resinovič, 1993, str. 277.

Na drugi strani tudi Oracle Form Builder vključuje nekatere že narejene komponente, ki pa niso del Oracle Headstart Designer-ja. Gre za menu, ki nam ponuja:

- orodno vrstico, ki jo lahko po potrebi priredimo za lastne potrebe,
- menu s standardnimi funkcijami (Datoteka, Urejanje, Okno, Pomoč, idr.),
- formo za zamenjavo gesla,
- drugo.

3.2 Analiza odvisnosti

Razvijalci in projektanti se pogosto srečujejo z vprašanji:

- Koliko dela (časa) je potrebno za izvedbo določene spremembe?
- Koliko tvegamo ob zamenjavi komponente v določeni fazi projekta?

Na ta vprašanja včasih znajo odgovoriti sami projektanti, ki imajo v glavi oblikovanje celotnega sistema oz. tisti, ki preiščejo celotno programsko kodo, da dobijo željene odgovore. Vedno pogosteje pa projektni vodja nima pravega odgovora na podobna vprašanja in mora sprejemati odločitve brez njih. Problem izhaja iz pomanjkanja podrobnih informacij o odvisnostih in vplivu med programskimi komponentami.

Da bi podobne težave čim bolj omilili, morda celo odpravili, nam Oracle Designer ponuja orodje za analizo odvisnosti. (ang. Dependency Analysis).
Gre za sposobnost ugotavljanja kje oz. v kateri aplikaciji je uporabljena določena komponenta oz. definicija (npr. PL/SQL knjižnice, menu, orodna vrstica, tabela, stolpec, drugo). Ta funkcionalnost se najbolj odraža v fazah vzdrževanja informacijskega sistema, nepogrešljiva pa je tudi v drugih fazah razvoja informacijskega sistema. Odvisnost predstavlja povezavo med dvema programskima objektoma. Obstajajo dve vrsti odvisnosti:
vse kar določen objekt uporablja,

- vsi objekti, ki uporabljajo izbrani objekt.

Oracle Designer nam torej omogoča iskanje informacij glede odvisnosti form, knjižnic, PL/SQL programske logike na strani strežnika in ostalih tipov datotek. Vse to je mogoče videti skozi prijazen grafični vmesnik, ki na pregleden način, skozi drevesno strukturo, daje vpogled o odvisnosti različnih objektov. Tako lahko hitro ugotovimo na katere objekte bo npr. vplivala sprememba določenega objekta. Analizo odvisnosti lahko naredimo tudi za objekte, ki niso bili narejeni z Oracle Designer-jem, pomembno je le, da so shranjeni v repozitoriju.

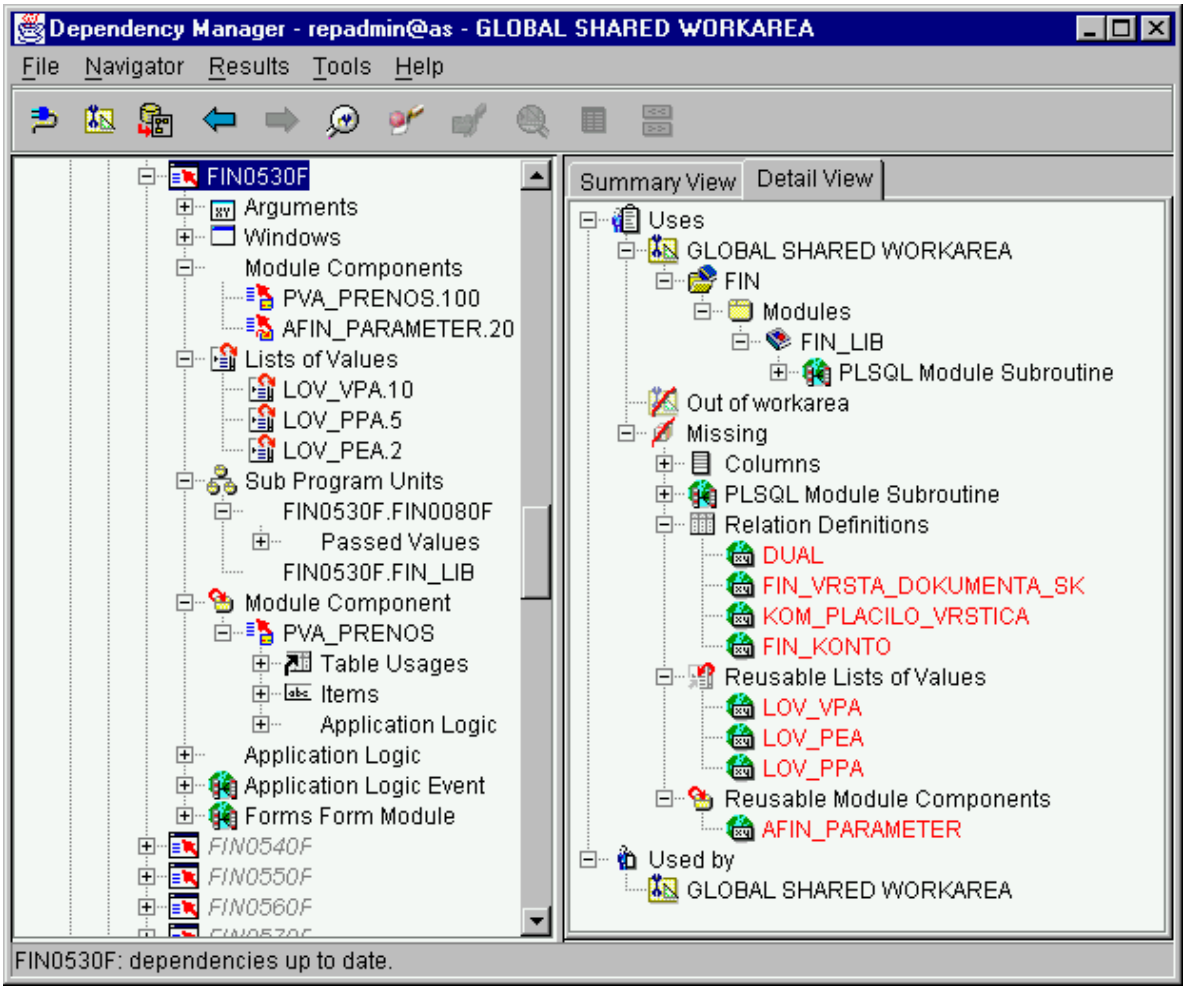
Tipi odvisnosti, ki jih lahko ugotovimo za zaslonski obrazec:

- uporaba tabel, vpogledov, stolpcev,
- priložene PL/SQL knjižnice,
- dedovane objekte iz objektne knjižnice,
- dedovane objekte iz drugih Oraclovih form,
- uporabljene oz. referencirane datoteke za ikone,
- klicane PL/SQL procedure in funkcije,
- uporaba oz. referenciranje menuja,
- drugo.

V levem delu okna (glej sliko 8, na str. 27) so prikazani vsi moduli delovnega področja repozitorija in njihovi sestavni deli (podatkovni bloki, liste vrednosti, klicani moduli, programska logika, idr.), v desnem delu pa so prikazane različne odvisnosti izbranega modula.

Orodje za analizo odvisnosti nedvomno izboljšuje sposobnost razvijalcev, tako velikih, kot manjših informacijskih sistemov, pri upravljanju razvoja in zagotavljanju večje kakovosti programov. Kompleksnejši kot je informacijski sistem, večje kot je število form, večje kot je število razvijalcev, bolj postane to orodje nepogrešljivo.

Slika 8: Analiza odvisnosti



Vir: Oracle Designer, 2002.

V orodju Oracle Developer te komponente ni, si pa lahko razvijalci pomagajo na več različnih načinov. Večino odvisnosti lahko ugotovimo iz drevesne strukture objektnega navigatorja. Morebitne povezave s pripetimi PL/SQL knjižnicami so razvidne iz veje Priložene knjižnice. Nekatere povezave se pokažejo pri prevajanju forme. Takrat se v drevesni strukturi objektnega navigatorja (veja Programske enote) pokažejo odvisnosti programske enote. Prikažejo se povezave z drugimi programskimi enotami, tabelami, vhodnimi in izhodnimi parametri, idr. Nekoliko težje je ugotoviti povezavo dveh različnih modulov (npr. ko iz forme kličemo izpis). Takšno odvisnost lahko ugotovimo z analizo programske logike, kar predstavlja časovno potratno nalogo v primerjavi z Oracle Designer-jem. Deloma lahko omenjene težave rešimo s kvalitetno sistemsko dokumentacijo.

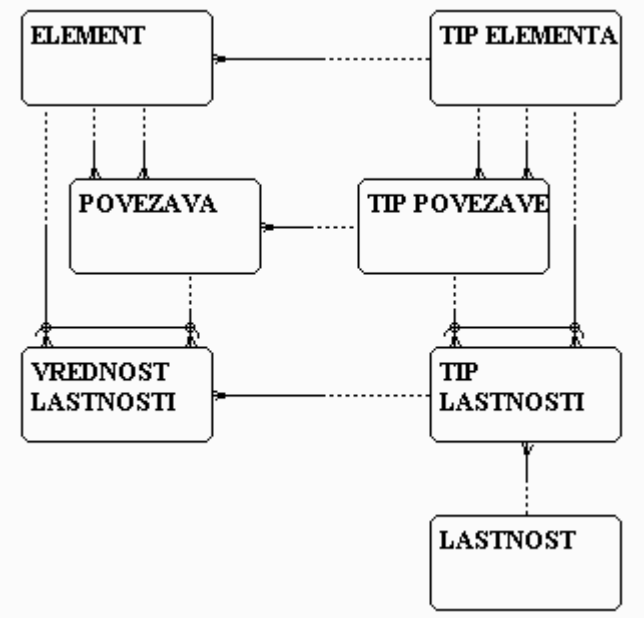
3.3 Sistemska dokumentacija

V orodju Oracle Designer nam je na voljo zbirka pomožnih programov, ki lajšajo administracijo repozitorija. To se nanaša na nameščanje, nadgrajevanje, vzdrževanje, pregledovanje repozitorija in njegovih elementov ter skrb za delovanje izpisov, ki producirajo sistemsko dokumentacijo. Za analizo in vzdrževanje vsebine repozitorija nam je tako na voljo okrog 100 že pripravljenih izpisov, z možnostjo izdelave lastnih izpisov v okolju Oracle Developer z orodjem Report Builder. Na podlagi teh izpisov Oracle Designer samodejno skrbi za celovitost in natančnost sistemske dokumentacije.
Tako postane izdelava sistemske dokumentacije avtomatizirana, standardizirana, lažja za vzdrževanje in dolgoročno cenejša. Največje prednosti se pokažejo v trenutkih, ko je potrebno zaradi spremembe v poslovnem okolju hitro vpeljati spremembo pri oblikovanju namenskih programov. Pri tem nas npr. zanima, kako bo sprememba na modulu vplivala na delovanje drugih modulov. Tu nam Oracle Designer priskoči na pomoč z zbirko že vnaprej pripravljenih izpisov za analizo vplivov vpeljave spremembe na sistemu (Atkins at al., 1999, str. 33).
Za kvalitetno sistemsko dokumentacijo in sistema samega je bistvenega pomena, da so vsi popravki na modulih shranjeni v repozitoriju, kajti le tako je možno informacije o spremembah zajeti s poizvedbami, ki so vsebovane v vnaprej pripravljenih izpisih.
Vsak izpis repozitorija prikazuje podatke le iz trenutnega delovnega področja. V primeru, ko ima delovno področje zunanje reference (objekt delovnega področja se navezuje na objekte izven njega), te niso vključene v izpis. Primer izpisa je priložen v prilogi 2.

3.4 Repozitorij Oracle Designer

Že večkrat omenjeni repozitorij je po mojem mnenju tisti gradnik Oracle Designer-ja, ki najbolj pripomore k večji produktivnosti razvijalcev in hkrati nedvomno predstavlja tudi največjo razliko v primerjavi z Oracle Form Builder-jem. V primerjavi z datotečnim sistemom, se tukaj vse informacije in podatki shranjujejo v sistemskih tabelah fizične podatkovne zbirke, katero lahko enostavno pregledujemo z SQL ukazi. Ravno zaradi tega Oracle Designer ponuja vrsto analitičnih pripomočkov (analiza odvisnosti, sistemska dokumentacija), s pomočjo katerih si lahko pomagamo v različnih fazah razvoja programske opreme.
Vsi grafični vmesniki, programski generatorji in uporabni programi v okviru obravnavanega programskega okolja operirajo s podatki v repozitoriju (za potrebe posameznih orodij so definirani različni vpogledi v podatke repozitorija.). Logična struktura repozitorija je sestavljena iz objektov lastnost (ang. property), tip lastnosti (ang. property type), vrednost lastnosti (ang. property value), element, tip elementa, povezava (ang. relation) in tip povezave (ang. relation type). Na sliki so prikazani osnovni elementi podatkovnega slovarja in povezave med njimi.

Slika 9: Zgradba repozitorija



Vir: Heričko, 2000.

Vsebina rapozitorija je predstavljena s pomočjo drevesne strukture, ki podpira funkcionalnost povleci in spusti. Izbrani objekt ali komponento označimo in prenesemo v grafični del, kjer ga lahko zatem obdelujemo in dopolnjujemo. Na podobnem principu (drevesna struktura in grafični del) je strukturiran tudi Oracle Form Builder-jev grafični vmesnik, z razliko, da imamo lahko v drevesni strukturi istočasno prikazanih le toliko modulov kot smo jih izbrali in odprli preko datotečnega sistema. Pri Oracle Designer-ju pa so istočasno dostopni vsi moduli določenega delovnega področja.
V okviru programskega okolja Oracle Designer je na voljo paket orodij, ki omogočajo delo z vsebino repozitorija. Omenjena orodja omogočajo razdelitev posameznih delov aplikacij med različnimi projekti in s tem ponovno uporabo elementov repozitorija. V okviru administracije repozitorija lahko prenašamo posamezne dele med različnimi repozitoriji za potrebe razdeljenega razvoja aplikacij.

3.5 Ponovna uporaba komponent

CASE repozitorij prav tako omogoča konceptu *ponovne uporabnosti*, da postane praktičen in uresničljiv. Ponovna uporaba je namreč lahko ključ za precejšen dvig produktivnosti. Seveda pa tukaj ne gre le za ponovno uporabo izvorne kode modulov, temveč tudi projektnih planov, prototipnih modelov, specifikacij (načrtovalskih), idr.

Definicija ponovne uporabe (Krmac Vatovec, 1997, str. 2): ponovna uporaba programske opreme je sposobnost razvoja novih aplikacij iz že obstoječe programske opreme.

Ponovno uporabna komponenta je katerakoli objekt, ki je bil načrtno razvit za uporabo, in tudi dejansko uporabljen, v več kot enem kontekstu. Lahko je to koda (npr. PL/SQL knjižnice), specifikacije zahtev in načrtovanja, procesi, metode, dokumentacija, testni primeri, celi sistemi ali podsistemi, modeli, vzorci, ogrodja, primerki objektov in drugo. Načeloma so ponovno uporabne komponente vsi možni izdelki, ki nastanejo v katerikoli fazi življenjskega cikla razvoja programske opreme.

Uspeh te metode je predvsem odvisen od navdušenja in sposobnosti programerjev za iskanje in uporabo ponovno uporabne programske opreme. Ponovna uporaba je omejena s kompleksnostjo opravila, saj je zelo težko vedeti, kaj je razpoložljivega in kako neko funkcijo ali modul uporabiti. Preden lahko določeno komponento ponovno uporabimo, moramo :

- komponento najti,
- vedeti njeno funkcionalnost, njen namen, in
- vedeti, kako to komponento ponovno uporabiti.

Pri vseh teh opravilih nam lahko pomaga dobra dokumentacija, kjer so vsi omenjeni vidiki podrobno opisani.

Zakaj za koncept ponovne uporabnost?

Dandanes so organizacije soočene z različnimi pritiski iz okolja, prisiljene so zmanjšati tako razvojni čas kakor čas plasiranja izdelka na tržišče, povečati morajo raznolikost ponujenih izdelkov, poleg tega pa povečati standardizacijo izdelkov. Ponovna uporaba predstavlja eno izmed sredstev za dosego teh ciljev. S krizo programske opreme, ko sta razvoj in vzdrževanje programske opreme predraga, postane potreba po povečanju produktivnosti razvoja realna potreba. S ponovno uporabo lahko bistveno pripomoremo k zadovoljevanju strankinih zahtev po večji kvaliteti in funkcionalnosti programske opreme.
Prednosti, ki jih ta tehnika prinaša, še niso popolnoma izkoriščene in realizirane. Pomagamo si lahko s primerno organizacijsko infrastrukturo in upravljanjem razvojnega procesa programske opreme. Z izvajanjem sistematične in učinkovite ponovne uporabe lahko dosežemo naslednje prednosti:

- nižji stroški bodočega vzdrževanja,
- hitrejši razvoj,
- večja kvaliteta,
- višja produktivnost.

Med prednosti pa lahko štejemo tudi dejstvo, da je programska oprema, ki je bila razvita za ponovno uporabo, načeloma dobro dokumentirana in testirana.

Zakaj proti konceptu ponovne uporabnosti?

Poleg pozitivnih lahko najdemo tudi negativne argumente ponovne uporabe. Nekatera zelo pogosta opravičila, razlogi, ki izražajo nenaklonjenost, predvsem razvijalcev/programerjev, nad ponovno uporabo, so (Krmac Vatovec, 1997, str. 4):

- Samo nesposobni posegajo po programski opremi, ki jo je ustvaril nekdo drug.
- Ponovna uporaba programske opreme izničuje sposobnost ustvarjanja programske opreme.
- Z vpeljavo ponovno uporabne programske opreme bom izgubil/a službo.
- Ponovno uporabna programska oprema ne more biti učinkovita.
- Nočem biti prvi/a.
- Poskušati ponovno uporabiti programsko opremo nekoga drugega je izguba časa.
- Ne verjamem, da je ponovna uporabnost programske opreme koncept, ki bo preživel.

Kot je razvidno iz zgornjih točk je človeški faktor ena izmed potencialnih ovir. Večji del profesionalnih razvijalcev programske opreme se bo raje lotilo pisanja programske opreme na novo, kakor da bi ponovno uporabilo že razvito (in testirano) programsko opremo, ki jo je izdelal oz. razvil nekdo drug. Ta problem se da delno rešiti z ustreznim posredovanjem vodstva.
Težje premostljiva je ovira pravne narave, do katere lahko pride pri pogodbeni programski opremi. V pogodbi, ki jo podpišeta proizvajalec in kupec programske opreme, navadno piše, da je programska oprema last kupca. V primeru, ko bi razvijalec v novem izdelku za novo stranko ponovno uporabil komponento, ki je del izdelka, katerega je že enkrat prodal drugi stranki, predstavlja to kršitev pogodbe.

Večina študij uspešnih programov ponovne uporabe, ki jih je opravila družba Technology Transfer International (TTI) iz Colorada na precejšnjem številu japonskih organizacij (tudi Fujitsu, Hitachi, Oki in Toshiba), je prišla do prepričljivih ugotovitev o izboljšanju produktivnosti kot posledice/rezultata sistematične ponovne uporabe :

- razvojni čas je bil skrajšan od 10 do 50 odstotkov,
- produktivnost je bila zvišana za 15 do 50 odstotkov,
- kvaliteta se je izboljšala za 20 do 35 odstotkov.

Ponovno uporabo programske opreme omogočata obe orodji, tako Oracle Designer-ju kot Form Builder, vendar je njena uporaba učinkovita le ob upoštevanju naslednjih predpostavk:

- komponente so vsem dostopne in znane,
- specifikacija in dokumentiranje ponovno uporabnih komponent,
- določena pravila ponovne uporabe,
- tesna povezanost in sodelovanje razvojne skupine,
- zavedanja prednosti ponovne uporabe.

Način ponovne uporabe programske opreme je zaradi strukture orodij in načina razvoja zaslonskih obrazcev nekoliko drugačen. V obeh lahko uporabljamo knjižnico objektov, z razliko v tem, da moramo v Oracle Designer-ju deklarativno specificirati ime knjižnice objektov in objekta, da lahko med generiranjem, na podlagi teh informacij, dodaja elemente iz knjižnice objektov v razvito aplikacijo. Ta način vpisovanja informacij o knjižnici objektov in objektih samih predstavlja zamudno nalogo, kajti to od razvijalca zahteva, da bodisi te informacije zna na pamet bodisi ima odprto objektno knjižnico, da lahko vidi imena objektov, katere namerava ponovno uporabiti. Lahko se tudi zgodi, da se razvijalec zmoti pri vnašanju/tipkanju imen objektne knjižnice in vsebovanih objektov.
V Form Builder-ju je to opravilo mnogo bolj enostavno. Pri aktiviranju orodja se objektne knjižnice naložijo same, sicer jih odpremo iz datotečnega sistema. Za ponovno uporabo pa je dovolj, da uporabimo tehniko povleci in spusti.
Po drugi strani ima Oracle Designer točno določen prostor v drevesni strukturi diagrama modulov, kamor shranjujemo ponovno uporabne komponente. S tem postane ponovna uporaba še enostavnejša, preglednejša in dostopnejša. Razvijalcem je ni treba iskati po datotečnem sistemu, kot je to potrebno pri Oracle Form Builder-ju, ker je zbrana na enem mestu. Pri gradnji forme in njenih gradnikov pa nam čarovniki nenehno ponujajo možnost ponovne uporabe.

Primer ponovne uporabe: Orodno vrstico v formi, ki jo je razvil in testiral en razvijalec, lahko uporabljajo pri razvoju svojih form tudi ostali razvijalci, namesto da bi jo oblikoval vsak zase.

3.6 Upravljanje z verzijami modulov

Kontrola verzij modulov je proces vzdrževanja večjega števila verzij razvitega objekta. Pregledovalec zgodovine verzij (ang. Version History Viewer) in pregledovalec dogodkov verzij (ang. Version Event Viewer) sta grafični orodji, ki jih ponuja Oracle Designer kot pripomočka za analizo zgodovine in zaporedij sprememb, ki so bile narejene na posameznem objektu. Kontrola verzij je omogočena tako za objekte, ki so shranjeni v repozitoriju, kot za tiste, ki so shranjeni na datotečnem sistemu.
Verzije objektov so predstavljene v drevesni strukturi s pomočjo vej, ki kažejo potek razvoja objektov. Vsaka verzija modula je identificirana z edinstveno številko (verzijo), ki jo generira sam sistem, lahko pa objekt identificiramo tudi z lastnimi oznakami.
Pregledovalec zgodovine verzij omogoča pregledovanje poteka razvoja določenega objekta. Verzije so med seboj povezane z asociacijami, katere hranijo informacije o spremembah glede na predhodno verzijo, med drugimi (Oracle Designer 6i, 2002):

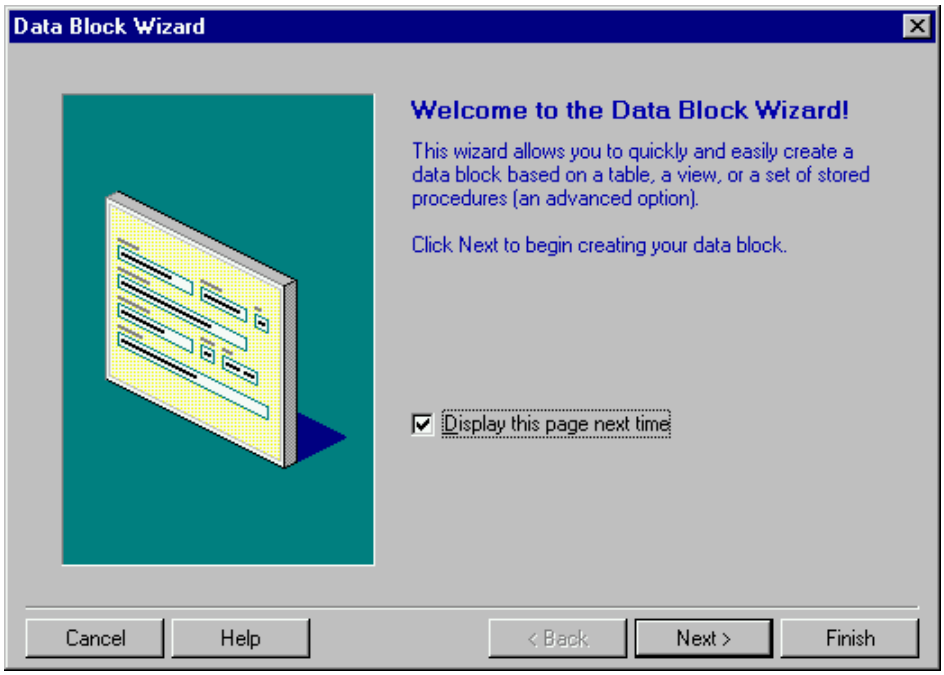
- edinstveno razpoznavno oznako verzije,
- uporabnika, ki je kreiral objekt,
- datum in čas kreacije objekta,
- uporabnika, ki je objekt spremenil,
- datum in čas spremembe,
- komentar (komentar uporabnika).

Da bi podatki o verzijah bili še bolj natančni je na voljo pregledovalec dogodkov verzij, ki omogoča pregled podrobnosti vseh dogodkov, ki so se zgodili na določenem objektu. Za vsak dogodek se v repozitorij shranijo informacije, vključno s tipom dogodka, verzijo, imenom objekta, datumom dogodka, uporabnikom, ki je dogodek začel in komentarjem o dogodku.
Verzije objektov je možno tudi medsebojno primerjati za analizo sprememb prehoda med različnimi verzijami objekta.
Funkcionalnost upravljanja z verzijami modulov ni samodejno omogočena ob namestitvi Oracle Designer-ja. Možnost kontrole verzij je torej opcijska in je zato prepuščena odločitvi posamezne razvojne skupine.

3.7 Čarovniki

Dandanes predstavljajo čarovniki sestavni del vseh boljših razvijalskih in drugih programskih orodij. Dejstvo je, da so dobro zasnovani čarovniki lahko v veliko pomoč, velikokrat pa se zgodi ravno obratno, da so nezaželeni, nadležni, predvsem pri tistih uporabnikih, ki program oz. orodje že dobra poznajo. V takih primerih je zaželeno, da ima orodje možnost onemogočenja čarovnika.
Zakaj omenjam čarovnike pri primerjavi Oracle Developer-ja in Oracle Designer-ja. Omenjam jih zato, ker so to komponente orodij, ki bistveno pripomorejo k hitrosti razvoja aplikacij in s tem prispevajo k večji učinkovitosti oz. produktivnosti razvijalca. Delo nam lajšajo predvsem pri rutinskih nalogah, npr. pri izdelavi preprostejših aplikacij, pri kreiranju objektov, pri izdelavi grafičnega vmesnika, nekateri pa zmorejo tudi kompleksnejše naloge. Na ta način lahko z vnosom nekaj podatkov in nekaj miškinimi potezami opravimo nalogo, za katero bi sicer potrebovali več časa. Velikokrat, s čarovnikom izdelane komponente, predstavljajo le izhodišče za kasnejšo nadgradnjo. Zelo so koristni v fazi spoznavanja in učenja orodja, poleg tega omogočajo hitro kreiranje aplikacij, čeprav enostavnih, ki jih lahko uporabimo kot izhodiščne prototipe za prikaz končnemu uporabniku. Pri Oraclu so pri izdelavi čarovnikov pazili na dobro mero, kar se nanaša na prijaznost do uporabnika, težavnost razumevanja, število korakov do končnega rezultata, možnost prekinitve dela s čarovnikom v katerikoli fazi kreiranja objekta, vračanje nazaj in naprej med koraki. Pri izgradnji posamezne komponente nas vodi korak za korakom, od začetka do konca izgradnje posamezne komponente, dopušča pa možnost kreiranja objekta tudi, če nismo dali skozi vseh predvidenih korakov. Če s produktom čarovnika nismo zadovoljni, nam je na voljo možnost ročnega manipuliranja lastnosti objekta. Lahko pa tak objekt zbrišemo in znova poženemo čarovnika za generiranje objekta. Na ta način bistveno skrajšajo čas razvijanja objektov forme.
Čarovnika sta sestavni del obeh orodij. Oracle Form Builder vključuje štiri čarovnike: za kreiranje podatkovnih blokov, za oblikovanje grafičnega vmesnika aplikacije, za kreiranje grafikonov in izdelavo list vrednosti.

Slika 10: **Primer čarovnika v Oracle Form Builder-ju**



Vir: Oracle Form Builder, 2002.

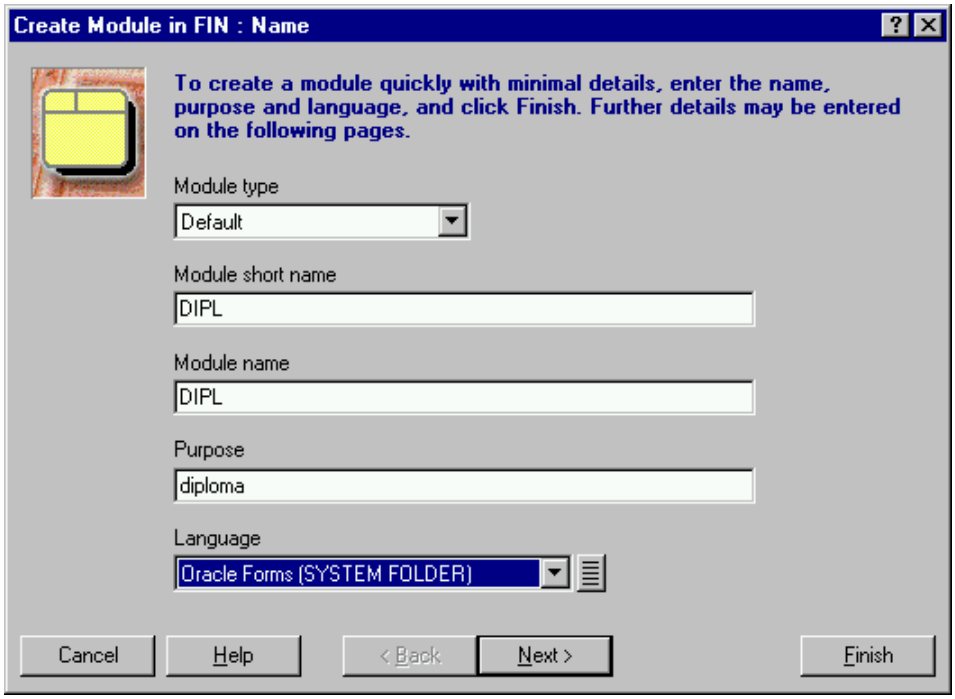
Ko je objekt kreiran lahko za isti objekt znova prikličemo čarovnika za morebitne spremembe ali popravke. To storimo tako, da se postavimo na objekt, katerega želimo modificirati in nato aktiviramo čarovnika, bodisi preko menuja, bodisi preko desnega gumba miške.

Pri Oracle Designer-ju so čarovniki številčnejši in zmogljivejši v primerjavi s čarovniki Oracle Form Builder-ja. Omogočajo kreiranje skoraj vseh objektov forme. Predpogoj za pravilno delovanje in popoln izkoristek nekaterih čarovnikov v Oracle Designer-ju je kvalitetno definiran podatkovni model, predvsem se to nanaša na relacije med posameznimi objekti v podatkovni strukturi (relacije med tabelami, glavni in tuji ključi). Na podlagi teh informacij čarovnik definira povezave med posameznim bloki aplikacije in nato sam kreira programsko logiko za usklajevanje glavnega in podrejenega bloka. Pri Oracle Form Builder-ju je to potrebno eksplicitno definirati med čarovnikovim postopkom kreiranja podatkovnih blokov oz. kasneje. Podobno kot Oracle Designer, tudi Oracle Form Builder kreira programsko logiko za usklajevanje glavnega in podrejenega bloka.

Čarovniki Oracle Designer-ja prispevajo k večji produktivnosti v primerjavi z Oracle Form Builder ne le zaradi večjega števila čarovnikov, temveč tudi zaradi večje uporabnosti končnega produkta čarovnika.

Eden takšnih je npr. čarovnik za klicane modulov, ki ga v Oracle Form Builder-ju ni. Omogoča kreiranje logike za klicanje drugih Oracle modulov (zaslonskih obrazcev, poročil, PL\SQL knjižnic). Predpogoj je, da so klicani moduli shranjeni v repozitoriju.

Slika 11: **Čarovnik za izdelavo novega modula v Oracle Designer-ju**



Vir: Oracle Designer, 2002.

3.8 Podpora skupinskemu delu

Značilnost vsakega (razen izjem) večjega projekta je sodelovanje večjega števila ljudi. Da bi bilo njihovo delo čim učinkovitejše je potrebno zagotoviti dobro komunikacijo in sodelovanje med njimi. Tukaj so nepogrešljivi sestanki na vseh nivojih, usklajevanje dela in drugi standardni prijemi, ki so se uveljavili skozi zgodovino.

Vse te značilnosti se pojavljajo tudi pri razvoju informacijskih sistemov, kjer pri oblikovanju skupne programske rešitve sodeluje večje število udeležencev, od svetovalcev, projektantov pa do razvijalcev. Pri razvoju informacijskega sistema prihaja do velikega števila interakcij med posameznimi področji (finance, kadri, proizvodnja), med programskimi komponentami, objekti in samimi podatki v bazi. Da bi vse te interakcije lažje obvladovali nam Designer-jev repozitorij omogoča optimalno rešitev. Informacijsko tehnologijo lahko uporabljamo tudi brez repozitorija, zlasti ko gre za enostavne programske rešitve, z le nekaj formami in izpisi. Njegova uporaba postane smiselna pri projektih z večjo stopnjo kompleksnosti. Repozitorij omogoča skupinam, da delajo na skupnih podatkih, procesih in definicijah modulov. Podpora skupinskemu delu, ki jo nudi vsem dostopen repozitorij, se izraža predvsem v konsistentnosti, zmanjšanju podvajanja, spodbujanju k ponovni uporabi, omogočanju večje kontrole in še bi lahko naštevali. Vse to lahko vpliva na hitrost razvoja aplikacij in celotnega informacijskega sistema.

Tako kot je možno posamezne objekte deliti z drugimi, jih je možno tudi zakleniti pred drugimi. S tem je še dodatno povečana kontrola nad objekti, istočasno pa to pomeni, da lahko

na repozitoriju dela večje število ljudi, ne da bi nas bilo strah, da bo kdo naše delo uničil ali kakorkoli nezaželeno spremenil. Vse to je mogoče doseči z dodeljevanjem pravic nad posameznimi objekti baze.

3.9 Razvoj zaslonskih obrazcev in oblikovanje grafičnega vmesnika aplikacije

Izdelava zaslonskih obrazcev je v obeh orodjih podobna. Obe uporabljata osnovna pogovorna okna (objektni/modul navigator, palete lastnosti, urejevalnik, PL/SQL Editor), podoben pa je tudi način oz. postopek razvoja form, če izvzamemo nastavitve za generator form in možnost vpliva na oblikovanje končnega grafičnega vmesnika aplikacije. To morda niti ne preseneča, saj z Oracle Designer-jem dejansko generiramo Developer aplikacije.
Oracle Form Builder omogoča ročno izdelavo vseh objektov in v končni fazi celotne aplikacije, kot alternativno možnost pa imamo na voljo čarovnike, ki omogočajo razvoj aplikacije praktično brez potrebe poznavanja programskega jezika. Seveda je na ta način mogoče kreirati le enostavnejše programske rešitve.
Na drugi strani nam Oracle Designer pri kreiranju vseh objektov pomaga z zmogljivimi čarovniki, ki bistveno olajšajo delo razvijalca. Ročno kreiranje je mišljeno zgolj kot prirejanje lastnosti objektov, ki smo jih kreirali s pomočjo čarovnikov.

Med najbolj opaznimi razlikami, za mnoge razvijalce tudi najšibkejša točka Oracle Designer-ja, je možnost vplivanja razvijalca na končni izgled grafičnega vmesnika programske rešitve. V dosedanjih verzijah so generatorji form pokazali omejitve, ki se nanašajo na kontrolo generiranja podobe uporabniškega vmesnika, zato je Oracle Form Builder na tem področju bližje razvijalcu, ker je veliko bolj prilagodljiv in dopušča oblikovanje izgleda brez večjih omejitev. Izgled forme oblikujemo sami s kreiranjem objektov, določanjem njihovih lastnosti in postavljanjem na platna. Objekte po platnu premikamo z miško oz. z ustreznimi tipkami in na ta način postavljamo objekte na željeno pozicijo. Zaslonski obrazec bo imel takšen izgled kot ga bomo sami oblikovali in videli v urejevalniku izgleda.
V nasprotju, generator form ne zna pozicijonirati generiranih objektov na določeno pozicijo (X,Y) v koordinatnem sistemu generiranega modula. Pozicijoniranje posameznih objektov poteka bolj na grobo, relativno glede na dimenzije vidnega dela platna. Končni izgled zaslonskega obrazca izdela na podlagi pravil, ki jih določimo s pomočjo generatorjevih prednostnih nastavitev in drugih deklarativnih pravil na nivoju postavk, skupin postavk, blokov, oken in forme. Razvijalec si lahko le približno predstavlja kakšen bo izgled, zelo težko pa določi natančno pozicijo postavk in drugih objektov na platnu. Izgleda zaslonskega obrazca namreč ne vidimo dokler ne generiramo forme. Odvisen bo predvsem od predhodno nastavljenih parametrov generiranja. Da bi v Oracle Designer-ju izdelali formo z zelo zahtevnim izgledom se lahko zgodi, da moramo formo (pre)večkrat generirati, preden bomo zadovoljni z izgledom. Dodaten problem pri generiranju lahko predstavlja zmogljivost strojne opreme. Manj zmogljivi računalniki (npr. Pentium 200, 128MB RAM) lahko za generiranje kompleksnejše forme potrebujejo tudi do 10 in več minut. Odvisno je tudi od zmogljivosti

računalnika, kjer je nameščen repozitorij. Kaj hitro se lahko zgodi, da zaradi izgleda forme izgubimo veliko dragocenega časa, kar za razvijalca in razvijalsko podjetje predstavlja nepovraten strošek. Določene prijeme pridobimo z izkušnjami, druge z ustreznim izobraževanjem, med razvijalci pa prevladuje mnenje, da moramo biti zadovoljni s tem, kar nam generator sam izdela/generira, sicer je bolje, da Oracle Designer-ja za razvoj zaslonskih obrazcev ne uporabljamo. Čas, ki bi ga porabili za oblikovanje grafičnega vmesnika, je po mojem mnenju bolje porabiti v prepričevanje naročnika, da izgled ni tako pomemben v primerjavi s funkcionalnostjo forme. Ena od možnih alternativ je ta, da generirano formo naknadno grafično obdelamo v Form Builder-ju. Problem je v tem, da te grafične spremembe ne bodo shranjene v repozitoriju in bodo z vsakim ponovnim generiranjem iz repozitorija izgubljene.

Z Oracle Designer-jem generirane aplikacije utegnejo imeti slabše odzivne čase v primerjavi z aplikacijami, ki so jih razvili izkušeni razvijalci v Oracle Form Builder-ju. Generatorji namreč generirajo veliko programske logike in nekateri njeni deli bodo lahko povsem odveč. Vzrok temu so obstoječe tehnike generiranja in generatorji, ki sami še niso dovolj dozoreli, da bi poleg združevanja deklarativnih definicij iz repozitorija, šablon, PL/SQL in objektnih knjižnic v eno celoto (delujočo formo), zmogli tudi samodejno optimizacijo generirane kode (Žabkar, 2000, str. 35).

Dvig učinkovitosti razvijalcev programskih rešitev ter večja zanesljivost delovanja aplikacij sta posledica dejstev, da orodje Oracle Designer avtomatizira delo, ki bi bilo sicer izpostavljeno možnim napakam in vedno enakim ponavljanjem postopkov. Prav tako govori v njegovo prid dejstvo, da je programska logika v modulih, generirana prek generatorja, kar pomeni, da je brez napak (velja tudi za Oracle Form Builder). To sicer ne pomeni, da bodo moduli resnično delovali tako kot je bilo opredeljeno v specifikaciji programa, kajti generator preverja le pravilnost programske logike. Posledično je kakovost generirananega sistema izboljšana, kar se odraža v krajšem času testiranja zaradi manjšega števila napak.

Sodeč po izkušnjah, ki jih imam z orodjema, sem mnenja, da so razvijalci produktivnejši pri delu z Oracle Designer-jem v primerih, ko gre za izgradnjo večjih oz. kompleksnejših informacijskih sistemov. Deloma se ta produktivnost nanaša tudi na področja (analiza, snovanje, vzdrževanje, idr.), ki jih v diplomskem delu nisem obravnaval.

4 SKLEP

Vsakdo, ki danes razvija programsko opremo ali dela kjerkoli na področju informatike, je soočen z nenehnimi spremembami. Vse izrazitejša konkurenčnost okolja in zmogljivejša informacijska tehnologija postavljajo pred razvijalska podjetja zahtevo po vse kakovostnejših in cenejših programskih rešitvah. Zato je za podjetja pomembno, da razumejo in se z rešitvami odzovejo na vse spremembe poslovanja, ki jih prinašajo novi gospodarski pogoji in razsežnosti. Prav tako je pomembno, da po(znajo) učinkovito uporabiti širok spekter orodij in tehnologij za vsakodnevno delo.

V času, ko se večina podjetij in panog srečuje z različnimi nihanji gospodarskega vzpona in recesije, lahko predstavlja nakup razvijalskega orodja pomemben korak za prihodnost in nadaljni razvoj podjetja. Zato je v rokah vodilnih mož podjetij za razvoj programskih rešitev pomembna odločitev, za katero orodje se bodo odločili.

Oracle Designer in Oracle Form Builder sta le dve izmed mnogih razvijalskih orodij, ki jih danes ponuja trg in dve izmed mnogih, ki jih ponuja Oracle Corporation. Za njuno primerjavo sem se odločil na podlagi izkušenj in namena poglobljenega spoznavanja strukture orodij ter možnosti, ki jih orodji nudita.

Orodji se razlikujeta v nekaterih pomembnih postavkah, ki so predvsem odraz arhitekture orodij. Kot je to značilno za večino CASE orodij, tudi CASE Oracle Designer temelji na repozitoriju. Le-ta predstavlja centralno skladišče definicij o vseh podatkih, ki so potrebne za izgradnjo informacijskega sistema. Zgrajen je na principu standardne Oraclove baze, kjer se v primerjavi z datotečnim sistemom, katerega uporablja Oracle Form Builder, vse informacije in podatki shranjujejo v sistemskih tabelah fizične podatkovne zbirke. Ta struktura omogoča Oracle Designer-ju uporabo vrsto analitičnih pripomočkov/orodij, ki jih pri Oracle Form Builder-ju prav zaradi tega ne najdemo.
Poleg repozitorija predstavlja večjo razliko med orodjema sam generator zaslonskih obrazcev, ki pa je še vedno nekoliko omejen, predvsem ko gre za oblikovanje in kontrolo generiranja podobe grafičnega vmesnika aplikacije. Tudi sam postopek generiranja form predstavlja kompleksno nalogo, ki zajema večje število korakov in od razvijalca zahteva daljšo dobo učenja.
Velik vtis so name naredili zmogljivi čarovniki Oracle Designer-ja, ki z avtomatizacijo rutinskih nalog bistveno vplivajo na večjo produktivnost razvijalcev, znižanje stroškov razvoja in večjo kakovost razvitega sistema.

V diplomi predstavljene razlike so po mojem mnenju tiste, ki najbolje odsevajo prednosti in slabosti posameznega orodja, hkrati pa vplivajo na delo razvijalcev, kar v končni fazi tudi najbolj šteje. Pri tem je potrebno upoštevati dejstvo, da se orodji še vedno razvijata in se temu primerno nenehno dopolnjujeta, zato nekaterih razlik oz. slabosti ne gre vzeti za stalnice,

temveč le kot hibo trenutne verzije orodja. Ena od posledic razvoja je tudi ta, da sta se orodji še pred kratkim prodajali ločeno, sedaj pa ju je Oracle ponudil v enem paketu (Oracle Developer Suite).

Skozi primerjavo orodij, prebiranjem literature in izkušenj z orodjema sem prišel do ugotovitve, da lahko dodatno vplivamo na večjo produktivnost razvoja, če orodji uporabljamo skupaj. Prav tako sem prišel do spoznanja, da splošne porasti produktivnosti razvijalca ne gre pripisati enemu samemu dejavniku, v tem primeru orodju Oracle Designer-ju, čeprav je le-ta zelo pomemben. Pomembna dejavnika, ki ravno tako odločilno vplivata na produktivnost razvijalca, sta motiviranost in kvalificiranost.

5 LITERATURA IN VIRI

LITERATURA:

1. Billings Chris, Billings Maria: Rapid Development with Oracle Designer/2000. Addison Wesley, 1996. 560 str.
2. Bulent Cinarkaya, Tushar Gadhia: Oracle Education, Developer/2000:Build Forms 1(2), Volume 1(2), Student Guide. januar 1998. 452 str.
3. Dai Clegg: Using Designer/2000 for Team-Based Rapid Application Development. Chertsey, Velika Britanija. 15 str.
4. Davelaar Steven: 100% Generation: No More Excuses**!,** Quality Management Services. Oracle Netherlands. str. 8.
5. Davelaar Steven, Luc van den Haegen, Muller Sandra: Headstart Oracle Designer 2.1.2 Template Package User Guide, verzija 1.2.1. Nizozemska: De Meern. marec 1999. 151 str.
6. Fred Bethke et al.: Oracle Forms Developer: Form Builder Reference, Release *6i,*Volume 1. januar 2000. 766 str.
7. Gradišar Miro, Resinovič Gortan: Osnove informatike. Ljubljana: Ekonomska fakulteta, 1993. 334 str.
8. Helen Barnes, Mark Pirie, Simon Stow: Designer/2000 for Oracle Forms Developers, An introduction to modelling and generation, An Oracle Technical White Paper. november 1995. 38 str.
9. Heričko Marjan et al.: Dnevi slovenske informatike, Sožitje modelirnih in implemetacijskih orodij. Portorož, 21.-24. april 1999, str. 10-18.
10. Heričko Marjan: Case. Maribor: Univerza v Mariboru, [URL: http://lisa.uni-mb.si/student/predmeti/ois/case.htm], 03.05.2002.
11. Hipsley Paul: Developing Client/Server applications with Oracle Developer/2000, Indianapolis: Sams Publishing Indianapolis. 1996. 640 str.
12. Jelka Cimerman: Izdelava aplikacije kadrovska evidenca z uporabo orodja Oracle Developer/2000. Diplomska naloga poslovne šole. Ljubljana: Ekonomska fakulteta, 1999. 29 str.
13. Koletzke Peter, Dorsey Paul: Oracle Designer Handbook. Second Edition. Berkeley, Osborn McGrow Hill, 1998. 610 str.
14. Lulushi Albert: Inside Oracle Designer/2000. Prentice Hall, 1997. 949 str.
15. MacCullough-Dieter, MacCullough Carol: Oracle8 developer's guide. Foster City, 1999. 559 str.
16. Madžarević Dario: Izkušnje pri uporabi metodologije Oracle CDM Advantage v projektih prenove informacijskih sistemov velikih družb, 6 strokovno srečanje uporabnikov programske opreme Oracle SIOUG 2001 skupaj s konferenco Oracle iWorld, Portorož, september 2001, str. 21.
17. Oracle Corporation: Using Oracle Repository for Software Configuration Management. An Oracle Technical White Paper. februar 2001. 16 str.

18. Oracle Corporation: Oracle Designer Features Overview. An Oracle White Paper, november 1998. 10 str.
19. Oracle Corporation: Oracle Designer/2000. Server Design and Generation. An Oracle White Paper, maj 1998. 26 str.
20. Oracle Corporation: Oracle CDM Advantage 2.0, Oracle Consulting. 4 str.
21. Oracle Corporation: Oracle Designer *6i* Product Overview. An Oracle Technical White Paper, oktober 2000. 13 str.
22. Oracle Corporation: Building Oracle Forms Applications Using Designer 6i. An Oracle Technical White Paper, October 2000. 15 str.
23. Palčič Rok, Požar Borut: Dnevi slovenske informatike: Kako smo izbrali orodja za razvoj poslovnih aplikacij, Portorož, 21.-24. april 1999, str. 19-25.
24. Turk Ivan et al: Pojmovnik poslovne informatike. Ljubljana: Društvo ekonomistov Ljubljana, 1978. 446 str.
25. Vatovec Krmac Evelin: Ponovna uporaba - Kaj, Zakaj, Kdaj In Kako. Maribor: Univerza v Mariboru, 1997. 12 str.
    [URL: http://lisa.uni-mb.si/cot/cotl/december97/ponovna_uporaba.htm]
26. Žabkar Andrej: Popolno generiranje programskih rešitev z orodjem CASE: primer Oracle Designer. Magistrsko delo. Ljubljana: Ekonomska fakulteta, september 2000. 88 str.

VIRI:

1. Atkins Ken: An Overview of Oracle Designer Generation. MS Power Point predstavitev, 1999.
2. Headstart Oracle Designer. De Meern, Nizozemska: White Paper, november 1998. 4 str.
3. Inroduction to Oracle Forms. Oracle Forms Developer's guide.
   [URL:http://technet.oracle.com/doc/forms45/f45dg/ch1.htm], 04.03.2002.
4. Oracle Designer 6i, sprotna pomoč, 2002.
5. Oracle Developer 6i, sprotna pomoč, 2002.
6. Oracle Developer/2000, Release 2.0 for Windows NT. 5 str.
   [URL:http://www.esprint.com/Software/Oracle/dev2.htm], 03.04.2002.

6 PRILOGA 1

Pojmovnik/slovar

API	(ang. Application Programming Interface) Programski vmesnik razvojenga orodja.
CASE	(ang. Computer-Aided Software Engineering) Računalniško podprto načrtovanje sistemov.
I-CASE	(ang. Integrated Computer-Aided Software Engineering) Integrirano računalniško podprto načrtovanje sistemov.
Orodje CASE	(ang. Case tool) Programsko orodje, ki avtomatizira (vsaj deloma) določeno nalogo življenjskega cikla programskega opreme.
GUI	(ang. Graphical User Interface) Grafični vmesnik aplikacije (npr. zaslonskega obrazca) za končnega uporabnika.
ODBC	(ang. Open Database Connectivity) Odprto povezovanje baz podatkov.
OLE	(ang. Object Linking and Embedding) Sklicevanje in uporaba obstoječega objekta.
SQL	(ang. Structured Query Language) Jezik za dostop, manipuliranje in kontrolo podatkov v bazi.
PL/SQL	(ang. Procedural Language/Structured Query Language) Proceduralni jezik – Strukturirani poizvedovalni jezik - podaljšek SQL-a.
BPR	(ang. Business Process Reengineering) Prenova poslovnih procesov.
RAD	(ang. Rapid Application Development) Metodologija, ki omogoča razvijalcem hiter razvoj delujočih aplikacij.
DDE	(ang. Dynamic Data Exchange) Dinamično izmenjevanje podatkov.

Odjemalec/strežnik (ang. Client/server)
Arhitektura v kateri strežnik streže odjemalcu zahtevane informacije in nudi razne »storitve«.

Čarovnik (ang. Wizard)

Objektni navigator (ang. Object Navigator)

Paleta lastnosti (ang. Property Palette)

PL/SQL urejevalnik (ang. PL/SQL Editor)

Urejevalnik izgleda (ang. Layout Editor/Layout Model)

Povleci in spusti (ang. Drag & Drop)
Tehnika, ki omogoča prenašanje objektov z miško na način povleci in spusti.

Glavni - podrejeni (ang. Master – detail)
Predstavlja razmerje dveh podatkovnih blokov v formi ali tabel v podatkovni bazi (1:N).

Vpogled (ang. View)
Objekt baze podatkov, ki temelji na eni ali več tabelah.

»Razhroščevalno okolje« (ang. Debugging environment)
Orodje, ki pregleduje pravilnost sintakse programske logike.

Lastnost ang. Property
Tip lastnosti ang. Property type
Vrednost lastnosti ang. Property value
Povezava ang. Relation
Tip povezave ang. Relation type

7 PRILOGA 2

Slika 12: **Primer izpisa iz repozitorija:**

Module Documentation Report

Container: FIN **Version:**

Module

Name : FIN0010F
Short Name : OBDOBJE KNJIŽENJA
Language : Oracle Forms
Purpose : Obrazec za vzdrževanje aktivnega obdobja knjiženja
Type : Default

User/Help Text

Obrazec je namenjen vzdrževanju aktivnega obdobja knjiženja. Vsa navedena obdobja so aktivna. Obdobje deaktiviramo, tako da ga brišemo iz tabele.

Component Name: OBK_VNOS **Type** : Specific Module Component

Detailed Table Usage: FIN OBDOBJE KNJIZENJA

Comment :

Data Bound Items

Item Name: ID **Column** : ID
Comment :

Item Name: OBDOBJE **Column** : OBDOBJE
Comment :

Item Name: DATSPR **Column** : DATSPR
Comment :

Item Name: DATVNO **Column** : DATVNO
Comment :

Item Name: UPOSPR **Column** : UPOSPR
Comment :

Vir: Oracle Designer, 2002.