PRAKTIČNE IZKUŠNJE V PROCESU REINŽENIRINGA

RADO JENSTERLE
GENIS d.o.o.
Tržaška 40, Ljubljana

Povzetek

Moderna metoda uporablja orodja CASE, upošteva predhodni obstoj aplikacij, predpostavlja relacijsko in ciljno okolje, tako da klasične faze dopolni, spremeni njihovo težišče, racionalizira tehnike posamezne faze in se predvsem poglobljeno ukvarja z osnovnim in globalnim dizajnom podatkovnih struktur in procesov. Zlasti ne zanemarja organizacijskega vidika informacijskega sistema. Dober projekt je simfonija projektnega vodenja, poslovnega znanja, tehnologije in človeške kreativnosti. Rezultat je privlačna in obvladljiva izvedba aplikativne rešitve, če ...

Abstract

Modern design uses CASE tools, takes into account the previous condition of applications, presupposes relational and objective environment so that it complements the classical development stages, changes their accent and rationalizes the techniques of individual stages. Above all it deals in detail with basic and global design of data structures and processes. Moreover, it does not neglect organizational issues of informations systems design. Thus, a good project is a symphony of project management, business knowledge, technology and human creativity. The result might be an attractive and user friendly application, if only...

Ključne besede: *reinženiring, izkušnje, metode*

1. Uvod

Klasična fazna metoda razvoja aplikacij upošteva naslednje faze:

- analiza in arhitektura sistema,
- posnetek in analiza stanja,
- logično snovanje podatkov in procesov,
- fizično snovanje podatkov in procesov,
- programiranje, testiranje in vrednotenje aplikacije,
- uvajanje, izvajanje in vzdrževanje sistema.

Proces reinženiringa je metodološko spremenjen postopek klasičnega razvoja informacijskih sistemov, razpet preko istih razvojnih faz. Tehnologija CASE (Computer Aided System Engineering) s CARE (Computer Aided Re Engineering) navdihom daje takt pristopu rekonstrukcije.

Kot primer vzemimo renovacijo aplikacije, ki je tesno povezana s svojim poslovnim okoljem. Zajema področje spremljanja poslovnih resursov, poslovnih dogodkov, kupce in dobavitelje s saldakonti do prometa in glavne knjige. Aplikacija delno vključuje tudi statistično obdelavo operativnih podatkov do nivoja taktičnega odločanja. Celotna podatkovna baza vsebuje nad 10.000.000 zapisov, zavzema nad 1,5 GB aktivnih podatkov, pričakuje se dnevno nad 10.000 transakcij z želenim časom odgovora okoli 1 sek, mesečno se izvajajo večdnevne serijske obdelave in reorganizacije. Topologija obdelave je centralizirana z možnostjo strežnik-odjemalec in je močno distribuirana po celi državi. Prvotna zasnova sistema je bila centralizirana z močnim poudarkom na serijski obdelavi in uporabo indeksnih in sekvencielnih datotek. V prvotnem sistemu je obstajalo nekaj 100 bolj ali manj aktivnih programov.

2. Analiza in arhitektura sistema

Ker je v igri reinženiring delnega informacijskega sistema, ni potrebno izvajati celotno arhitekturo sistema. Potrebno je revidirati poslovne cilje in strategijo poslovnega sistema ali podsistema, ki bo vključevala nove vidike informacijske podpore, za kar je potrebno najti odgovorno osebo, formulirati temeljne poslovne izjave ključnih managerjev in analizirati bistvene poslovne probleme, želje in zahteve odgovornih končnih uporabnikov. Poudarek je na besedi 'bistvene', ker nikakor ni potrebno v tem primeru zgubljati tedne ali celo mesece za to fazo.

Detajlnost snemanja starega stanja je odvisna predvsem od radikalnosti bodočih sprememb, torej predvsem od strahu pred rizikom prehoda iz starega v novi sistem. Detajlni posnetek nam kasneje omogoča natannejše planiranje samega prehoda v novi sistem.

3. Posnetek in analiza stanja

Analiza stanja aplikativnega sistema za reinženiring se bistveno razlikuje od analize stanja pri klasičnem razvoju. Samo stanje je precej bolje poznano in transparentno, saj je aplikacija že informacijsko podprta. Ni potrebno izgubljati veliko časa z indentifikacijo posameznih podatkovnih in procesnih elementov, dobimo jih iz starega sistema in jih potem samo še oplemenitimo. S pomočjo orodij CARE se ta postopek skrajša na nekaj dni ali tednov, odvisno od količine in raznovrstnosti (VSAM, DL/I, SQL/DS, DB2, DB2/2, dBASE...) izvornega okolja.

Izdelava takega orodja lahko traja nekaj tednov, vendar je orodje potem večkrat uporabno. Rezultat posnetka je shranjen v repozitoriju orodja CASE, kar dodatno bogati celoten postopek. Ko s takim orodjem razpolagamo, lahko v nekaj dneh posnamemo na stotine zapisov in na tisoče polj z njihovimi definicijami in redefinicijami vred. Vse te podatkovne strukture lahko prikažemo grafično z njihovimi medsebojnimi povezavami in smiselno skozi repozitorij orodja CASE. Kasnejša analiza elementov se najbolj produktivno izvaja ravno s pomočjo repozitorija.

4. Logično snovanje podatkov in procesov

Osnovni cilj te faze je, da s pomočjo orodij CASE detajlno izrazimo elemente podatkovne strukture in procesov informacijskega sistema v duhu njegove arhitekture, pri čemer tehnike snovanja temeljijo na entitetno-relacijskih izhodiščih (metodi Entity-Relationship eXtended oz. E-RX in/ali Relational Data Modelling oz. RDM) ter prototipiranju informacijskih rešitev za kontrolo procesov.

Bistvena razlika glede na klasičen razvoj je v spremenjenem težišču same metode. V procesu reinženiringa za delni informacijski sistem (naš primer) ni nujno potrebno izdelati model E-RX, kajti prej opisani postopek posnetka starega stanja nam omogoča na osnovi grafike direktno izdelavo modela RDM, ki dodatno močno upošteva lepe zmožnosti repozitorija orodja CASE .

S tem se bistveno skrajša čas izdelave in čas kasnejšega vzdrževanja projektne dokumentacije. Nekoliko zamaknjena paralelna izdelava procesnih prototipov s pomočjo orodja CASE nam služi kot verifikacija pomenov in strukture podatkov, ki je lahko ob primerni organiziranosti projekta tudi temeljna uporabnikova potrditev pravilnosti našega dela.

V okolju CICS ali PC-WINDOWS večina programerske kode pripada delu z zaslonskimi maskami in dialogi. Zato je zelo težko natančno standardizirati mejo med funkcijami, ki bi jih radi zajeli v funkcionalni model procesov, in tistimi, ki jih ni potrebno. Funkcionalni model (Data Flow Diagram oz. DFD in podobno) ne bi smel biti obremenjen z neposlovnimi detajli (z detajli, ki izvirajo iz informacijskega okolja). Prototip procesov v primerni kombinaciji z podatkovnim modelom v veliki meri (zlasti v slučaju reinženiringa) zadovoljuje logično snovanje sistema. Nekateri tak pristop imenujejo 'data run' metoda (izvirnik CSAR limited Canada). V našem primeru je snovanje tudi metoda za finaliziranje stanja poslovnega sistema. To pomeni, da se je del metode po zaslugi nove tehnologije, ki je klasično pripadal fazi posnetek stanja, premaknil v fazo logičnega snovanja. Ko sta podatkovni model in pripadajoči prototip v zaključni delovni verziji, nastopi tudi najugodnejši trenutek za dokončno verifikacijo arhitekture, ki jo lahko potrdimo na posebej organiziranem sestanku z najvišjimi odgovornimi managerji. V tem momentu so spremembe še vedno možne, potrebni čas za izvajanje sprememb pa se meri v dnevih.

Izkušnje kažejo, da je podatkovni model, ki obsega 50 entitet z 1000 podatkovnimi elementi, možno 90% dokončati v 1 do 2 mesecih. Variacija časa je odvisna predvsem od nivoja detajlnosti (90%) in teksta dokumentacije. Ocena 90% je popolnoma prikladna, saj potrebni čas za večjo gotovost v tej fazi raste eksponencialno, marginalni dodatki in popravki v kasnejših fazah pa nam bodo odvzeli komaj nekaj dni.

Za prototip je potrebno polovica časa, ki je bil potrošen za podatkovno modeliranje. Čas za izdelavo prototipa je med drugim odvisen od standardov izgledov zaslonskih mask in dialogov in se bistveno skrajša, če je prototipno orodje povezano z generatorjem izvorne kode.

5. Fizično snovanje podatkov in procesov

Velika pridobitev tehnologije CASE so orodja za (pol)-avtomatsko izvajanje raznih faz. Tako je s pomočjo podatkovnega modela RDM možno direktno pripraviti sheme za ciljni RSUPB (relacijski sistem za upravljanje s podatkovno bazo). Izbiramo lahko med več kot 16 različnimi relacijskimi okolji za različne platforme (HW, SW, sistemska in aplikativna podlaga računalnika PC, UNIX ali HOST). Današnja orodja CASE nam v tej fazi omogočajo celo performančno usklajevanje tabel in indeksov. Postopek vključno z normalizacijo traja od nekaj minut do največ nekaj ur, kar pomeni enostavnejše vzdrževanje projektne dokumentacije ob raznih spremembah.

Fizično snovanje procesov je omejeno na fizično uvajanje prototipne aplikacije v bodoče okolje. Z aplikacijskimi generatorji je možno v nekaj dneh (ali celo v enem dnevu, če je prototipno okolje integrirano z aplikacijskim generatorjem) predstaviti dialoge zaslonskih mask v produkcijskem okolju. Očitno je učinkovitost odvisna od izbranega okolja, vsekakor pa je to fazo možno izvesti v nekaj dneh (tednih) in ne mesecih kot nekoč.

Bistvo tehnologije CASE v tej fazi je prilagodljivost orodja do sprememb, ki se v tej fazi izvajajo skozi celoten razvoj (manjkajočih 10%).

6. Programiranje, testiranje in vrednotenje aplikacije

Izvedbena faza je še vedno od vseh najbolj kočljiva, pa čeprav jo radi uvrščamo med obrtniške aktivnosti. Ob slabem pristopu k programiranju s testiranjem lahko izgubimo

tu največ dragocenega časa. Z nepremišljeno organizacijo z nevestnimi uporabniki in slabimi programerji lahko izgubimo mesece ali celo polletja. Praksa je pokazala, da so programerji tudi najmanj motivirani za dovzemanje sprememb na področju tehnologije.

Menim, da je pravilen izbor programsko-produkcijskega okolja strateškega pomena za časovno obvladovanje vsakega reinženiringa. Predlagam najmanj uporabo visoko standardiziranih templatov (programskih vzorcev), veliko bolj pa generatorjev programske izvorne kode. Tako bomo skrajšali čas programiranja, testiranje bo kvalitetnejše, vzdrževanje poenoteno in neodvisno od avtorjev programov. Taki generatorji razpolagajo tudi z možnostjo aktivnega prototipiranja, kar pomeni pomemben stik s končnim uporabnikom in dokončno vrednotenje aplikacije. Eden ali dva programerja iz tima lahko na koncu obdržita funkcijo aplikativnih vzdrževalcev.

V potreben izvedbeni čas je pomembno vračunati tudi efekte samega reinženiringa. Dejstvo, da govorimo o reinženiringu, nam govori v prid skrajšanju in kvalitetnejši izvedbi predhodnih faz. V svetu so v glavnem opustili zamisel o reinženiringu stare kode v novo kodo, razen ko ekonomika zaradi velikega števila programov nadomesti rizik, ali ko je tak prehod možen zaradi srečnih okoliščin sorodnosti starega in novega okolja. Zelo pomembno je, da se zavedamo, da so uporabniki v starem sistemu navajeni na uporabo npr. 250 programov, medtem ko bi jih ob klasičnem razvoju izdelali kvečjemu 150, medtem ko bi ostalih 100 izdelali kasneje skozi več let. Običajno se v procesu reinženiringa sama aplikacija dodatno funkcionalno razširi za 30 do 50%. Čas programiranja v reinženiringu je lahko krajši izključno na račun boljše tehnologije in organizacije, dejansko pa je zaradi števila programskih enot daljši kot bi bil v klasičnem razvoju.

V idealnih pogojih bi v našem primeru izvedbeno fazo lahko zaključili v 3 mesecih do pol leta. Pri tem je poleg tehnologije in organizacije pomembna izkušenost programerjev, uigranost programerskega tima in seveda število razpoložljivih programerjev. 80% programov je običajno programersko enostavnih in jih je možno izdelovati paralelno z večjim številom programerjev in celo z zunanjimi sodelavci. 15% programov je bolj zapletene narave in predstavljajo kritično pot v časovnem planiranju. Vedno moramo računati s kakimi 5 do 10 izjemno kompleksnimi programi, ki jih programiramo, popravljamo in izboljšujemo skoraj celoten čas razvoja. Obstaja tudi kakšen program, ki ni nikoli dokončan in se stalno spreminja še potem, ko je v produkciji. Medtem ko se stroški prvih faz gibljejo predvsem na račun tehnologije in strokovnosti, se v tej fazi izrazito prevesijo na račun organizacije in delovne sile. Investicija v dober generator izvorne kode, ki je morda še povezan z okoljem CASE, se vsekakor hitro obrestuje. Tak generator pa je tudi eden od učinkovitih načinov kako zlomiti pasivni odpor starejših programerjev do novosti v svetu informatike in uvesti učinkovite programerske standarde.

7. Uvajanje, izvajanje in vzdrževanje aplikacije

Najpomembnejši del te faze je priprava in izdelava orodja za prehod iz starega v novo okolje. Običajno gre za prehod iz indeksno naravnanega okolja v relacijsko okolje, iz serijsko poudarjene produkcije v interaktivno produkcijo, iz centralizirane topologije v distribuirano topologijo ter iz homogene platforme v heterogeno platformo. Kompleksnost je odvisna od stopnje svobode bodočega sistema. Današnja orodja I-CASE (Integrated CASE, ki združuje orodja za snovanje in izvedbo) in izkušenost nam omogočajo enoten metodološki pristop v vseh plasteh raznolikosti. Bistveni del priprave na prehod se nanaša na pretvorbo ključev starih zapisov v nove zapise in izdelavo ustreznih orodij. Potrebno je stare zapise izvleči iz starega sistema, preurediti ključe na novi sistem, dodati nove vrednosti za novo nastala polja in zagotoviti referencijalno integriteto med ključi. Ker stari sistemi niso povsem sledili referencijalni integriteti in ker se vedno v starih podatkih nahajajo napake, je potrebno izgraditi ustrezno orodje. Potrebno je izgraditi razna orodja za splošno kontrolo raznih napak starega sistema.

Očitno je največ dela ravno z izdelavo raznih orodij. Izkušnje kažejo, da bi tako orodje za naš zgoraj omenjeni primer vsebovalo nekaj 100 korakov porazdeljenih na nekaj 10 postopkov. Ocena je zelo pogojna, izdelava takega orodja pa lahko zahteva od enega človeka več mesecev dela. Orodja je možno začeti izdelovati takoj po končani fazi fizičnega snovanja, torej vzporedno s programiranjem. To pomeni, da izdelava takega orodja ni na kritični poti projekta ter obstaja dovolj časa za kvalitetno izdelavo in testiranje. Orodje lahko izdela samo strokovnjak, ki popolnoma obvlada postopke snovanja sistemov in ciljni RSUPB z vsemi performančnimi detajli. Poznavanje izvornega sistema ni nujno.

Za obvladanje rizika prehoda predlagam večfazen prehod po lokacijah in/ali podsistemih, pri čemer naj bo prva faza prehoda podatkovno čim manjša. Tak način omogoča postopno uvajanje uporabnikov in enostavno dokončno preverjanje aplikacije in orodij. Zavedati se moramo, da je uporabnik navajen na delovanje starega sistema in da je njegova strpnost manjša kot v primeru uvajanja povsem nove aplikacije. V tem momentu se pokažejo vsi konflikti, ki so bili eventuelno prikriti, zato so kvalitetno vodenje, diplomacija in ustrezna predpriprava uporabnika temelj zmanjševanja rizika prehoda. Dolžina ene faze prehoda naj smiselno soupada s periodičnimi obdelavami (npr. mesečnimi), kar služi za dodatno kontrolo točnosti prenosov podatkov, saj se morata periodični obdelavi na starem in na novem sistemu ujemati. Odvisno od dolžine posamezne faze prehoda in njihovega števila se sam postopek prehoda lahko raztegne čez več mesecev. Bistveno skrajševanje ni možno, saj izobraževanje in uvajanje uporabnikov zahtevata svoj čas in organizacijski napor, omejena razpoložljivost uvajalcev pa zmanjšuje možnosti hkratnega delovanja. Kljub vsemu prehod iz starega sistema v novi ne spada v kritične faze reinženiringa.

Postopka izvajanja in vzdrževanja se v odnosu na klasičen razvoj poenostavita, saj so bile vse dosedanje faze izvedene kvalitetneje in podprte z vrsto orodij.

Če naš projekt predvideva distribuirano topologijo in večplatformsko rešitev (npr. strežnik-odjemalec ali več enakovrednih sistemov), potem je treba prišteti še čas za performančno usklajevanje distribuirane podatkovne baze, razširitev standardov za programiranje in izdelavo postopkov za povezovanje lokacij in različnih tehnologij. To so tako imenovane planirane posebnosti novega sistema.

8. Zaključek

Uporaba repozitorija v orodju CASE nam omogoča kvalitetno povezovanje in vključevanje v podsisteme v drugih poslovnih okoljih in na drugačnih tehnoloških platformah. Nujno je treba razmišljati o posebni funkciji upravljanja z repozitorijem in podatkovnimi strukturami, ki nam omogoča širjenje prek meja podsistema.

Ves postopek reinženiringa v okvirih zgoraj navedenega primera bi lahko izvedli v obdobju 1 leta (9 do 15 mesecev plus planirane posebnosti). Glavni del osnovne investicije gre na račun opreme in delovne sile, planirane posebnosti pa na račun kompleksnosti lahko privedejo tudi do posebnih stroškov. Večina posebnosti, ki se pojavljajo sproti kot nenapovedljive situacije ali slučajna presenečenja, se lahko sproti rešuje v okviru določenega planiranega procenta rizika.

Če za reinženiring obstajajo realne in poslovne podmene, potem danes ni več toliko tehnični podvig, kakor predvsem spopad notranjih sil, ki mejijo na območja informacijske in poslovne kulture.

■

Navodila avtorjem

Prispevke pošiljajte v predpisani obliki na naslov Slovensko društvo Informatika, 61000 Ljubljana, Vožarski pot 12, s pripisom za revijo Uporabna informatika.

Če je možno, naj bo članek lektoriran. V uredništvu bomo opravili korekturo in se po presoji posvetovali z avtorjem, da članek tudi lektoriramo.

Prispevek naj bo v obsegu največ avtorska pola (30.000 znakov) za strokovne članke in približno 2 do 3 tiskane strani za druge prispevke. Vsak strokovni članek naj ima na začetku povzetek v slovenskem in v angleškem jeziku.

Pošljite ga na disketi in odtisnjenega na papirju. Napisan je lahko v kateremkoli urejevalniku besedil, vendar naj bo na disketi tudi kopija v ASCII formatu. Na disketi označite, kateri urejevalnik ste uporabili, in ime datoteke. Datoteko imenujte s svojim priimkom, n. pr. Novak. doc ali Novak.txt.

Slike, ki ste jih izdelali z grafičnim programom, označite podobno. Na natisnjenem izvodu članka naj bo jasno vidno, kam sodi posamezna slika. Lahko priložite tudi originalne predloge, ki jih na hrbtni strani označite s številkami, tako kot v natisnjenem besedilu.

Pišite v razmaku vrstic 1, brez posebnih ali poudarjenih črk ali podčrtovanja, za ločilom na koncu stavka napravite samo en prazen prostor, ne uporabljajte zamika pri odstavkih.

Za vsa vprašanja se obračajte na tehnično urednico Katarino Puc, 61000 Ljubljana, Ulica Gubčeve brigade, tel. 1271-579, elektronska pošta Katarina.Puc@uni-lj.si