UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

UROŠ GREGORIČ

UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

UPORABA METODE ŽIVLJENJSKEGA CIKLA PRI RAZVOJU PROGRAMSKE REŠITVE

Ljubljana, julij 2003 UROŠ GREGORIČ

IZJAVA

Študent Uroš Gregorič izjavljam, da sem avtor tega diplomskega dela, ki sem ga napisal pod mentorstvom prof. dr. Jaklič Jurija in dovolim objavo diplomskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne _____ Podpis: _____

Priloge:

1. Uvod

1.1. Opis problematike

V zadnjem obdobju se je število golf igrišč v Sloveniji izredno hitro povečevalo. Posledično se je povečalo število igralcev in igralk. Za razvoj tekmovalnega športa skrbi Golf zveza Slovenije. Tekmovalne ambicije njenih igralcev postajajo vse večje. Za doseganje vrhunskih rezultatov si danes v večini športov pomagajo z računalniško analizo in statistikami. Slovenski igralci za svojo analizo igre do sedaj še ne uporabljajo računalniško podprte tehnologije. Tisti, ki vodijo lastno statistiko, jo še vedno zapisujejo v papirni obliki. Z novo programsko rešitvijo se bo tako čas potreben za vnos podatkov skrajšal, hkrati pa bodo igralci dobili več informacij o svoji igri.

Cilj diplomske naloge je razviti programsko rešitev, ki bo igralcem in igralkam golfa omogočila lažje analiziranje treningov in turnirjev. Na ta način bodo igralci dobili povratno informacijo o učinkovitosti njihovega treninga. Hkrati bodo analize v pomoč tudi trenerjem in profesionalnim učiteljem, ki skrbijo za oblikovanje treningov. Programska rešitev bo namenjena predvsem vrhunskim igralcem golfa, saj jim bo dajala izredno podrobno statistiko njihovih turnirjev in treningov na igrišču in v telovadnici. Tudi ostali igralci bodo lahko uporabljali novo programsko rešitev. Vsak igralec bo glede na svoje potrebe uporabljal različne analize.

Za razvoj informacijskega sistema bo uporabljena metoda življenjskega cikla. Ta predvideva razvoj programske rešitve v petih fazah. Za to metodo sem se odločil, zaradi sistematičnosti poteka dela pri razvoju in izdelavi informacijskega sistema. Z dobrim prehodnim planiranjem informacijskega sistema je kasnejše delo programerju močno olajšano in končna programska rešitev je zanesljivejša.

1.2. Namen in vsebina dela

Z diplomsko nalogo želim znanje, ki sem ga dobil pri predmetih poslovno-informacijske smeri, uporabiti na praktičnem primeru. Odločil sem se, da na podlagi metode življenjskega cikla razvijem informacijski sistem, ki bo zajemal in analiziral rezultate turnirjev in treninge vrhunskih igralcev golfa.

Diplomska naloga je sestavljena iz več delov. V grobem jo lahko razdelimo samo na dva. Prvi del je bolj teoretski, in se navezuje na znanja iz baz podatkov in razvoja informacijskih sistemov. Opisani so posamezni pristopi razvoja informacijskega sistema, največ poudarka pa

je na metodi življenjskega cikla. V nadaljevanju je opisano okolje delovanja interneta, intraneta in ekstraneta. V današnjem poslovnem okolju je varnost podatkov še kako pomembna, zato je posebno poglavje namenjeno varnosti informacijskega sistema.

Drugi del diplomske naloge je vezan na razvoj informacijskega sistema za Golf zvezo Slovenije. V petih fazah so razložene bistvene značilnosti razvoja, v vsaki je na koncu opis izdelave faze za končnega naročnika. Prva faza se navezuje na načrtovanje informacijskega sistema, kjer so pojasnjeni razlogi za uvedbo informacijskega sistema in izvedena je študija izvedljivosti. V drugi fazi, imenovani analiza informacijskega sistema, so podrobno analizirane potrebe uporabnikov in sedanje stanje. S pomočjo ankete, v kateri je sodelovalo 53 amaterskih in profesionalnih igralcev golfa, in viri na internetu so oblikovane zahteve informacijskega sistema. Na koncu druge faze je prišla odločitev, za katero okolje začeti razvijati informacijski sistem. Potrebne je bila izbira med informacijskim sistemom za osebni računalnik, programom na internetu ali dlančniku. Na podlagi tabele prednosti in slabosti ter rezultatov anket, je bila optimalna odločitev za razvoj informacijskega sistema za osebni računalnik. V tretji fazi, to je zasnova informacijskega sistema, je potrebno logični model prevesti v fizični. Izdelan je diagram entitet-povezav in hkrati tudi fizični model. Oblikovati je bilo potrebno vnosne forme in vse potrebne izpise. Naslednja faza predvideva razvoj informacijskega sistema. Na tem mestu programer na podlagi prejšnjih faz napiše programsko kodo. Za dobro delovanje sistema so potrebna testiranja delovanja sistema. Testiranje poteka v testnem okolju. Če želimo, da uporabniki lahko uporabljajo nov informacijski sistem, jih je potrebno ustrezno izobraziti in jim pokazati prednosti novega sistema. V zadnji fazi, ki teče dokler se informacijski sistem uporablja, je potrebno informacijski sistem vzdrževati. Količina vzdrževanja je odvisna od kontrole med posameznimi fazami in pravočasnosti odkrivanja napak.

2. Informacijski sistem

S pomočjo informacijskega sistema zbiramo, obdelujemo, shranjujemo, analiziramo in uporabljamo informacije. Tako kot vsak sistem tudi informacijski sistem vključuje vhodne elemente (podatke) in izhodne elemente (poročila, kalkulacije). Iz obdelanih podatkov izdela poročila, ki so poslana uporabnikom ali drugim sistemom. Informacijski sistem deluje znotraj poslovnega okolja. Ločimo formalne in neformalne informacijske sisteme. Prvi vključujejo vnaprej dogovorjene postopke, standardizirane vhodne in izhodne elemente, torej so dokaj fiksno definirani. Drugi se nahajajo v različnih oblikah. Pomembno je, da se zavedamo tudi neformalnih informacijskih sistemov. Ravno slednji lahko vsebujejo informacijske vire, ki včasih predstavlja vmesni del med formalnim in neformalnim informacijskim sistemom (Turban et al., 1999, str. 17-18) .

Informacijski sistem, je sistem v katerem se ustvarjajo, shranjujejo in pretakajo informacije. S pomočjo informacijskih sistemov premostimo problem transformacije podatkov, ter prostorske in časovne pregrade. Pri transformaciji podatkov se srečamo z dvema problemoma, prvi predstavlja izbiro postopkov, drugi vključuje problem izvedbe postopkov za ustvarjanje informacij ob uporabi različnih tehnik. Osnovne funkcije informacijskih sistemov se izvajajo na prostorsko različnih lokacijah, zato je potrebno zagotoviti prenos podatkov med različnimi lokacijami. To naredi komunikacijska oprema, ki včasih samo pretvarja podatke v obliko, ki je primerna za prenos, nadzoruje in krmili promet s podatki. Podatke praviloma ne obdelujemo in uporabljamo v trenutku nastanka, zato jih je potrebno shraniti na različne nosilce podatkov. Tako shranjeni podatki so na voljo za kasnejšo uporabo (Gradišar et al., 2001, str. 338).

2.1. Baza podatkov

Baza podatkov je urejena zbirka medsebojno povezanih podatkov, ki je shranjena na računalniškem nosilcu podatkov. Iz definicije lahko razberemo, da zapisi v papirni obliki ne predstavljajo baze podatkov, saj niso shranjeni v računalniku (Jaklič, 2003, str. 70). Sistem, s katerim upravljamo podatkovne baze (SUPB, angl. database management system – DBMS), je zbirka programov, s katerimi lahko izvajamo branje, pisanje, spreminjanje, brisanje in iskanje podatkov. SUPB, kot množica programov, je namenjena različnim uporabnikom, od načrtovalcev podatkovne baze, programerjev, skrbnikov baz podatkov do končnih uporabnikov. Z uporabo SUPB, je zagotovljena neodvisnost med programi in podatki. Uporabniki ne morejo neposredno dosegati ali spreminjati podatkov v bazi. Do podatkov lahko pridemo samo preko sistemov za upravljanje baz podatkov, ki napravi spremembo ali opravi želeno poizvedbo in nam vrne odgovor. Neodvisnost je zagotovljena s podatkovnim slovarjem, kjer je shranjen opis podatkovne baze, kot je struktura in tipi podatkov (Jaklič, 2003, str. 75-76).

SUPB je sestavljen iz petih glavnih komponent (Shelly et al., 1998, poglavje 8, str. 22):

- Jezik za definiranje podatkov, (angl. DDL data definition language) z njim je zapisana struktura baze podatkov. V shemi je definiran celotni podatkovni model, ki vključuje vsa polja, zapise in zvezo med tabelami. S pomočjo DDL lahko določimo tudi podsheme oziroma podatkovne baze posameznih uporabnikov.
- Jezik za manipulacijo podatkov (angl. DML data manipulation language) zagotovi potrebne ukaze za vse operacije v podatkovni bazi, na primer shranjevanje, urejanje, spreminjanje in brisanje zapisov.
- Poizvedbeni jezik (angl. query language) je nepostopkovni jezik, ki se uporablja za dostop do baze podatkov. Nepostopkovni jezik pomeni, da uporabnik določi rezultat, kako do tega rezultata pride ni pomembno. Najbolj poznan in razširjen jezik za delo s podatkovnimi

bazami je SQL (angl. structured query language), ki je omogočen na skoraj vseh platformah. Na tem mestu lahko opozorim, da je delo z osnovnimi poizvedbami s pomočjo SQL jezika izredno preprosto, vendar pa je lahko hkrati tudi omejujoče, predvsem pri zahtevnih poizvedbah.

- Podatkovni slovar (angl. data dictionary) služi kot centralno skladišče podatkov o podatkovni bazi. Vse sheme in podsheme so shranjene v podatkovnem slovarju.
- Uporabniški programi (angl. utilities programs). Večina SUPB ima že vključeno potrebno podporo za vzajemno obdelovanje podatkov, varnost, izdelavo rezervnih kopij, nadzor in neokrnjenost podatkov.

Bazo podatkov načrtujemo in zgradimo z določenim namenom, zato moramo označiti realni pogled na zunanje okolje. Vsaka sprememba v okolju zahteva spremembe tudi v podatkovni bazi. Dobro zgrajena baza podatkov je integrirana, kar pomeni, da vsebuje podatke za različne uporabnike. Praviloma posamezni uporabnik ne dostopa do vseh podatkov. Pri izgradnje baze podatkov moramo biti izredno pozorni na redundanco oziroma na podvajanje podatkov. Posamezen podatek, naj se zapiše na čim manj mestih, tako pospešimo poizvedbe in spreminjanje podatkov. Baze podatkov so neodvisne od programov, ki jih bodo uporabljali uporabniki. To omogoči večjo prilagodljivost pri razvoju informacijskega sistema, saj lahko dodajamo nove programske rešitve, ne da bi to vplivalo na delo že obstoječih informacijskih sistemov (Jaklič, 2003, str. 70-71).

Baza podatkov lahko za podjetje predstavlja osnovni vir sredstev, tako kot ostala sredstva na primer zaloga materiala ali poslovni prostori. Da je njena vloga uspešna, pa mora zadovoljiti naslednje kriterije (Gradišar et al., 2001, str. 222-224):

- Prilagodljivost pomeni, da lahko sistem razširimo, spremenimo ali skrčimo, s tem pa ne ogrozimo njegovega delovanja. Baza podatkov mora biti organizirana tako, da omogoča njeno uporabo, ki ni bila predvidena v samem načrtovanju, vendar so okoliščine pokazale, da je sprememba nujna.
- Dostop do podatkov je omogočen različnim uporabnikom znotraj organizacije. Posamezni uporabniki vidijo podatke v želenih oblikah.
- Točnost zagotovimo tako, da podatke ob shranjevanju preverimo. S to kontrolo odpravimo določene napake, vseh pa s kontrolo ni mogoče odpraviti.
- Učinkovitost je pomembna tudi pri bazi podatkov. Učinkovitost je odvisna od načina organizacije podatkov v računalniku. Neorganiziranost podatkov upočasnjuje postopke.
- Popolnost podatkov lahko zagotovimo v primeru, da se izognemo nepotrebni redundanci. Ker je podatek zapisan samo v eni bazi podatkov, podatkov ni potrebno podvajati.
- Podatki kot vir sredstev zahtevajo ustrezno zaščito. Varnost zagotovimo tako, da preprečimo nepooblaščenim osebam dostop do podatkov. To največkrat naredimo z gesli.

2.2. Sestavni deli informacijskega sistema

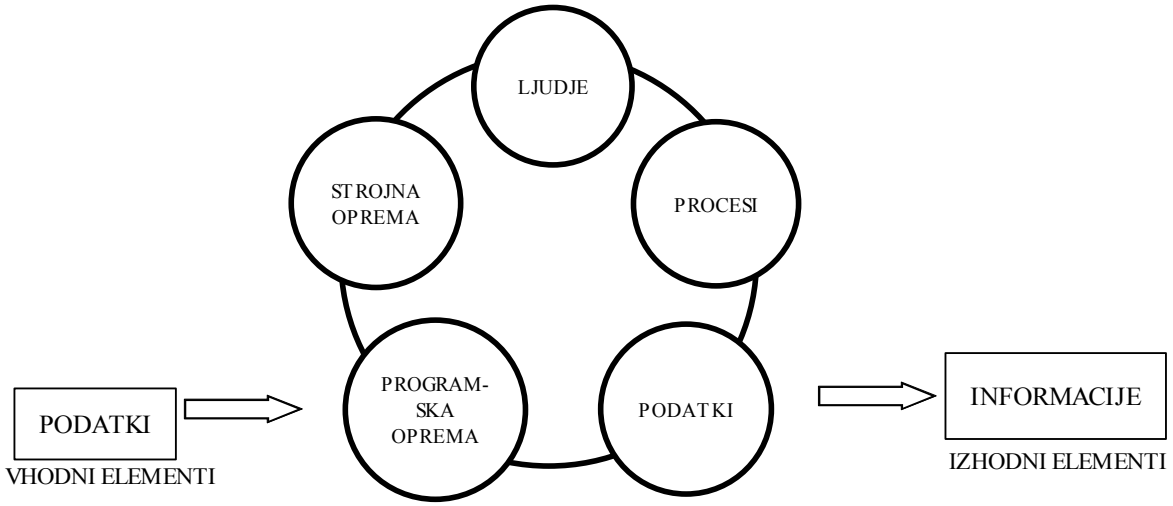
Posamezni avtorji poimenujejo sestavne dele informacijskega sistema različno. Kljub tem razlikam, pa lahko rečem, da imajo podobne lastnosti. V vsak sistem vstopajo vhodni elementi na primer podatki. V poslovnem procesu se ti podatki uporabijo in preoblikujejo in dobimo izhodne elemente, imenovane informacije. V tem procesu pa sodeluje tudi pet elementov, slika 2. (Shelly et al, 1998, poglavje 1, str. 4-6)

Informacijski sistem je učinkovita uporaba programske opreme, strojne opreme, podatkov, procesov in ljudi z namenom, da dosežemo želen rezultat. Informacijski sistem lahko imenujemo tudi krajše sistem.

Strojna oprema se navezuje na fizično opremo sistema. Ta vključuje računalnike, računalniška omrežja, komunikacijsko opremo, tiskalnike, skenerje, digitalne naprave za beleženje podatkov in ostalo tehnološko infrastrukturo.

Programska oprema je sestavljena iz sistemske programske opreme in aplikacijske programske opreme. Sistemska programska oprema nadzoruje strojno opremo in okolje programske opreme. Sistemska programska oprema vsebuje operacijski sistem, komunikacijsko programsko opremo in javne programčke, ki opravljajo osnovne funkcije kot razvrščanje datotek, spreminjanje formatov, izdelava varnostnih kopij in podobno. Sistemska programska oprema omogoča uporabnikom dostop do lokalnih mrež, interneta, intraneta. Aplikacijska programska oprema je sestavljena iz programov, ki obdelujejo podatke, ko uporabnik želi določeno informacijo.

Slika 2: Sestavni deli informacijskega sistema



Vir: Shelly et al.,1998 , poglavje 1, str. 5.

Podatki, ki so shranjeni v datotekah in bazah podatkov, so osnovna komponenta vsakega informacijskega sistema. Informacije so lahko pridobljene direktno iz podatkovnih baz ali pa so sestavljene iz posameznih podatkov.

Procesi definirajo opravila, ki jih morajo opraviti zaposleni, vključujoč uporabnike, managerje in sistemske analitike. Procesi so največkrat zapisani v pisni ali računalniški obliki.

Osnovni namen informacijskega sistema je zagotovitev koristnih informacij. Uporabniki informacijskega sistema so lahko zaposleni, stranke, dobavitelji in ostali, ki direktno vstopajo v poslovni sistem. Informacijski sistemi vključujejo tudi sistemske analitike, programerje in IS managerje.

2.3. Vrste informacijskih sistemov

Informacijske sisteme lahko razvrstimo v šest velikih skupin, izvajalni informacijski sistem, informacijski sistem za upravljanje, sistemi za podporo odločanju, informacijski sistemi za direktorje, ekspertni sistemi, sistemi za avtomatizacijo pisarniškega dela (Shelly et al., 1998, poglavje 1, str.10), Gradišar in Resinovič dodata še sisteme za podporo dela v skupini.

Izvajalni informacijski sistem (angl. operational systems ali transaction processing systems) obdeluje podatke zbrane v dnevnih poslovnih transakcijah podjetja. Izvajalni informacijski sistemi so bili prvi računalniško podprti, njihov namen je bil nadomestiti ročno delo z računalniško obdelavo in tako pospešiti procese, znižati stroške in povečati kakovost storitev (Shelly et al., 1998, poglavje 1, str.10). So dobro strukturirani ter temeljijo na podrobnih obdelavah poslovnih dokumentov (Gradišar et al., 2001, str. 368).

Informacijski sistemi za upravljanje (angl. management information systems) so računalniško podprti sistemi, ki zagotavljajo pravočasne in natančne informacije za vrhnje, srednje in nižje vodstvo. S temi informacijami lahko vodstvo sprejema boljše odločitve, kar pripomore k učinkovitejšemu poslovanju. Osnovna naloga informacijskih sistemov za upravljanje je osredotočenje na informacije, ki jih potrebujejo managerji. Informacijski sistem za upravljanje je lahko uporaben tudi na operativnem nivoju, kadar zaposleni potrebujejo informacije(Shelly et al., 1998, poglavje 1, str.10).

Pogosto managerji potrebujejo informacije, ki niso že vnaprej pripravljene. V ta namen so razvili posebno skupino informacijskih sistemov, imenovano sistemi za podporo odločanju (angl. desicion support systems). Ti sistemi pomagajo pri odločitvi z analizo zunanjih in notranjih podatkov. Notranje podatke pridobijo iz svojih datotek, kot so prodaja, proizvodnja in nabava. Zunanji podatki lahko vključujejo obrestno mero, populacijska gibanja in devizne

tečaje. Sistemi za podporo odločanju pogosto vključujejo poizvedbe, statistične analize, razpredelnice in grafe, ki pomagajo uporabnikom oceniti vse vhodne podatke. Naprednejši sistemi omogočajo uporabnikom, da sami ustvarijo svoj model (Shelly et al., 1998, poglavje 1, str.10-11). Sistemi so namenjeni reševanju slabo strukturiranih problemov (Gradišar et al., 2001, str. 371-372).

Veliko managerjevih odločitev je periodičnih in predvidljivih. Za te odločitve lahko potrebne informacije pripravimo vnaprej, tudi alternativne odločitve je možno določiti vnaprej. Te odločitve imenujemo strukturirane odločitve, saj se pojavljajo zaporedoma, imajo vnaprej določene informacijske zahteve in rezultat lahko predvidimo. Po drugi strani so odločitve, ki jih ne moremo predvideti. Nestrukturirane odločitve so zelo pogoste v vrhnjem managementu.

Informacijski sistemi za direktorje (angl. executive information system) zagotavljajo potrebne informacije za vrhnji management v primeru nestrukturiranih vprašanj. Informacijski sistem za direktorje združuje funkcije in sposobnosti sistemov za podporo odločanju in informacijskih sistemov za upravljanje, vendar z veliko fleksibilnosti pri reševanju nestrukturiranih problemov. Slabost teh sistemov je visoka cena, kar zavira množično uporabo v podjetjih (Shelly et al., 1998, poglavje 1, str. 11).

Ekspertni sistemi nadomeščajo človeško razmišljanje in odločanje, s kombiniranjem znanja strokovnjakov. Čeprav izgleda na prvi pogled, da resnično razmišljajo, je to omejeno na programske omejitve. Ti sistemi ne morejo delovati po zdravi pameti ali se odločati na podlagi intuicije. Ekspertni sistemi uporabljajo tehniko umetne inteligence, ki je aplikacija človeške inteligence v računalniški sistem. Ekspertni sistemi se uporabljajo v medicini, pri analizi geoloških podatkov ali analizi delovanja avtomobilskega motorja (Shelly et al., 1998, poglavje 1, str. 11).

V preteklosti so pisarniške sisteme (angl. office automation system) imenovali sistemi za avtomatizacijo pisarniškega dela, ker so bili prepričani, da bodo s temi sistemi zmanjšali število zaposlenih v pisarnah. Čeprav se je določeno delo zmanjšalo, pa se je število zaposlenih v pisarnah povečalo. Danes ima večina podjetij zmogljive pisarniške sisteme, ki vključujejo lokalno in širše omrežje, elektronsko pošto, video konference, grafične predstavitve, intranet in internet dostop zunaj podjetja. Pisarniški sistemi so pogosto oblikovani v stranka/strežnik arhitekturi, ki omogoči strankam dostop do podatkov v okviru celotnega podjetja (Shelly et al., 1998, poglavje 1, str. 11).

Sistemi za podporo dela v skupini (angl. groupware systems) prestavljajo računalniški paket, ki pomaga pri organizaciji dela skupine ljudi. Osnovni namen teh sistemov je izboljšati usklajevanje dela v skupinah. Skupinsko delo je v podjetjih vedno bolj pogosto, zato ti sistemi dobivajo na pomembnosti (Gradišar et al., 2001, str. 375-376).

3. Razvoj informacijskega sistema

Pri razvoju informacijskega sistema sodelujejo različni profili zaposlenih. Managerji določijo okvir informacijskega sistema, ta pa mora biti usklajen s strateškimi in taktičnimi cilji podjetja. Managerji spremljajo razvoj informacijskega sistema in pomagajo pri reševanju morebitnih težav. Uporabniki opredelijo sistemske zahteve do najmanjših podrobnosti. Ti podatki lahko natančno opredelijo vsebino informacijskega sistema. Informatiki pri razvoju izberejo ustrezno računalniško in komunikacijsko tehnologijo, oblikujejo bazo podatkov ter izdelajo računalniške aplikacije. Osnovni namen je izboljšati poslovanje podjetja. Da lahko vsi uspešno sodelujejo pri razvoju sistema se morajo razumeti in vsaj delno poznati vsa področja. Uporabniki in managerji morajo znati določiti zahteve na način razumljiv informatiku. Po drugi strani pa mora informatik vsaj okvirno razumeti delovanje podjetja. Posebej dobro mora poznati tisti del podjetja, za katerega razvija informacijski sistem. Vsi morajo poznati metode razvoja informacijskih sistemov.

Razvoj informacijskega sistema je posledica poslovnih problemov ali priložnosti, ki jo nekdo vidi. Naslednji korak je razvoj sistema. Ko je sistem vzpostavljen v podjetju je potrebno vzdrževanje, kar pomeni tudi prilaganje novim spremembam v okolju. Ko je potreba po spremembah velika, sistema ni več smotrno spreminjati, zato ga je potrebno zamenjati z novim. Splošni model tako vsebuje štiri osnovne faze v razvoju informacijskega sistema (Gradišar et al., 2001, str. 423):

- začetek,
- razvoj,
- uvajanje,
- izvajanje in vzdrževanje.

Te štiri faze vsebujejo vse metode. Intenzivnost posameznih faz pa se od metode do metode razlikuje.

3.1. Metoda življenjskega cikla

V današnjem dinamičnem okolju so konstantne spremembe nekaj normalnega. Z rastjo podjetja se spreminjajo tudi informacijske zahteve. Če želimo biti uspešni, se mora podjetje stalno prilagajati spremembam, to pa se odrazi tudi na informacijskem sistemu.

Metoda življenjskega cikla (angl. systems development life cycle) je sestavljena iz petih faz, ki jih v podjetju potrebujejo pri razvoju informacijskega sistema. Metoda je sestavljena iz naslednjih faz (Shelly et al., 1998, poglavje 1, str. 16-17):

- načrtovanje informacijskega sistema,
- analiza informacijskega sistema,
- zasnova informacijskega sistema,
- razvoj informacijskega sistema,
- vzdrževanje informacijskega sistema.

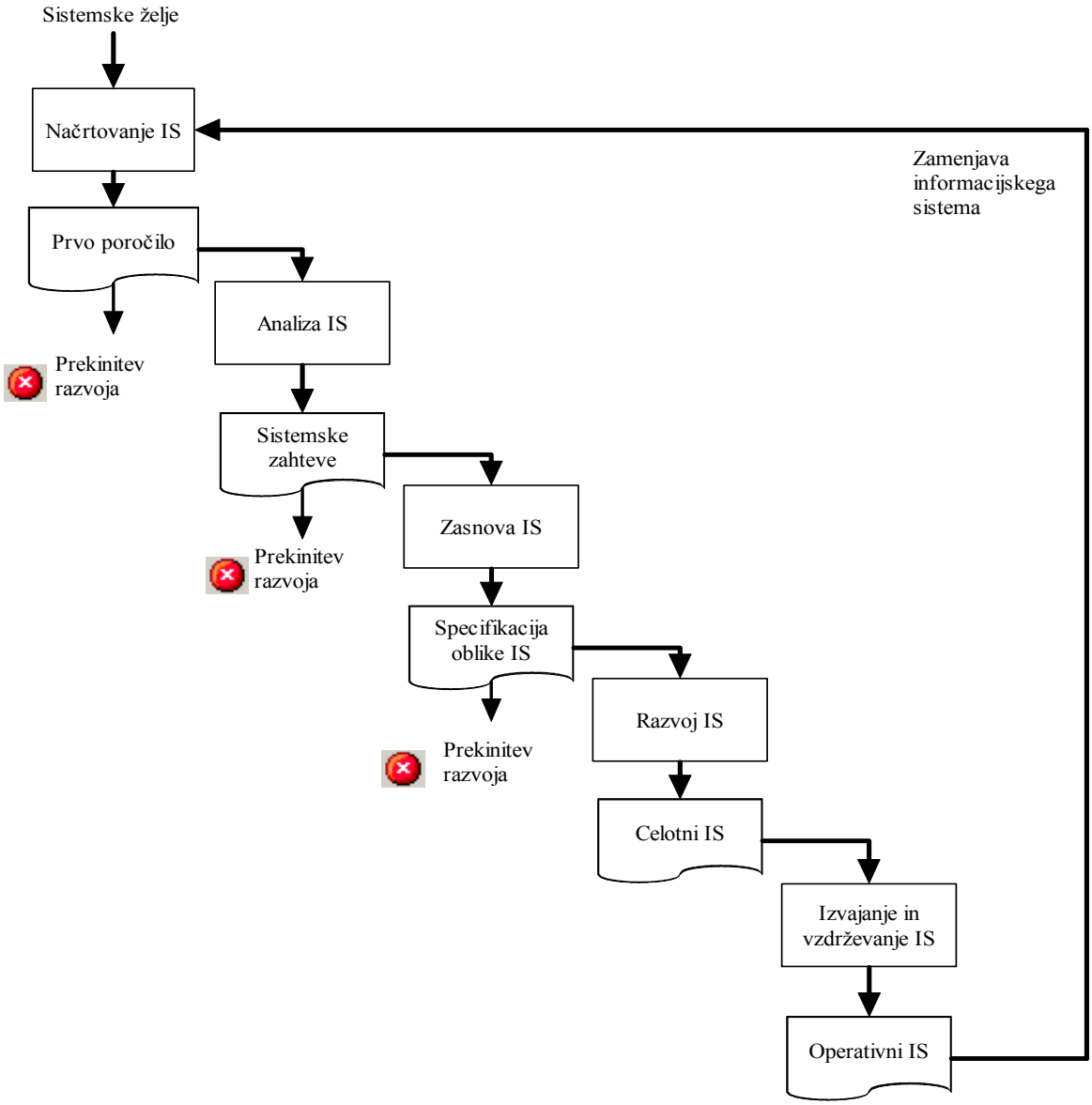
Vseh pet faz lahko grafično prikažemo na različne načine. Prvi način je kot razvojni krog, kjer si posamezne faze sledijo zaporedno. Drugi način je prikazan na sliki 3 imenovan slap (angl. watterfall model), kjer si posamezne faze nadaljujejo zaporedno. V vsakem koraku so prikazane vhodne in izhodne enote določene faze in možne odločitve. Informacijski sistem ni statičen, zato lahko pričakujemo, da se pri razvoju vračamo v predhodne faze.

Metoda življenjskega cikla je klasična metoda razvoja informacijskega sistema in se uporablja predvsem za razvoj večjih sistemov. Razvoj poteka fazno, v realnosti se faze med seboj pokrivajo ali pa se vračamo nazaj. Pri razvoju so pomembne vse faze. Z vidika stroškov pa so kritične prve faze, saj kasnejše vračanje nazaj pomeni dodatne stroške pri razvoju informacijskega sistema (Shelly et al., 1998, poglavje 1, str. 17-19).

Razvoj sistema se običajno začne s pisno željo po spremembi. V tem poročilu so opisane zahtevane spremembe ali izboljšave. Spremembe so lahko izredno majhne ali pa zelo drastične. Pri slednjih bo verjetno potrebno zamenjati stari informacijski sistem z novim. Namen te faze je ugotoviti naravo in velikost problema. Rezultat te faze imenujemo preliminarno poizvedbeno poročilo (angl. preliminary investigation report). Ta dokument je izredno pomemben, saj na podlagi njega potekajo vse nadaljnje faze. Sistemski analitik mora na tem mestu ugotoviti, za kakšne spremembe gre in ali je potrebno nadaljevati vse naslednje faze. Prvo fazo pogosto imenujemo tudi študija izvedljivosti (angl. Feasibility study), ki temelji na ekonomski, tehnični in operativni izvedljivosti.

Namen analize informacijskega sistema, je ugotoviti natančno, kako sistem deluje. Tako lahko določimo nove sistemske lastnosti in alternativne rešitve. Na tem mestu zbiramo podatke, z različnimi tehnikami, in kot rezultat dobimo sistemske zahteve (angl. system requirements document). V njem so razložene vse vodstvene in uporabniške zahteve, alternativne rešitve, stroški in naša priporočila. Tudi v tej fazi se lahko mangement odloči, da bo prekinil z razvojem sistema.

Slika 3: Pet faz pri metodi življenjskega cikla in končni rezultati



Vir: Shelly et al., 1998, poglavje 1, str. 17.

V fazi zasnove informacijskega sistema razvijemo sistemsko obliko, ki zadovolji vse sistemske zahteve. Med to fazo razvijemo logični model, kaj naj bi sistem izvajal in kako bo to izvajal. Sistemski analitik določi potrebne vhodne in izhodne enote, datoteke, aplikacijske programe in ročne postopke. Prav tako mora zagotoviti sistem, ki bo točen, zanesljiv, prilagodljiv in varen za uporabo. Specifikacija oblike informacijskega sistema (angl. system design specification) je rezultat tretje faze, ki pa ga mora potrditi vodstvo in uporabniki.

V četrti fazi, razvoj informacijskega sistema (angl. system implementation), se razvije informacijski sistem. Tu se programske rešitve napiše, testira in dokumentira. V primeru, da kupimo programski paket, je potrebno narediti določene prilagoditve sistema. Namen te faze

je razviti popolno delujoč informacijski sistem, ki mora biti pregledan in potrjen. Na koncu te faze je informacijski sistem pripravljen za uporabo. Na koncu je potrebno še dodati podatke, usposobiti nove uporabnike in pripraviti prehod iz starega sistema na novega.

Po začetnem uvajanju podjetje uporablja informacijski sistem za svoje poslovanje. V tej fazi sta potrebna vzdrževanje in posodobitev sistema. Vzdrževanje pomeni odpravljanje napak ali prilagoditve zunanjim zahtevam. Posodobitve so spremembe, ki omogočajo boljšo izrabo sistema. Nekateri informacijski sistemi so v uporabi več let, vendar se po nekaj letih uporabe večina sistemov zamenja z novimi. Tako se krog v metodi življenjskega cikla zaključi.

Kadar uporabimo metodo življenjskega cikla se srečamo z naslednjimi težavami. Veliko razvojnih projektov propade, čeprav uporabljajo metodo življenjskega cikla. Poslovno okolje je zelo različno od okolja izpred tridesetih let. Podjetja se morajo prilagajati spremembam veliko hitreje. Informacijska tehnologija je veliko zmogljivejša in vključuje uporabniške vmesnike, internet, stranka/strežnik arhitekturo, kar zahteva različen pristop k razvoju informacijskega sistema. Ali bi zaradi tega morali opustiti metodo življenjskega cikla? Mislim da ne. Še vedno ima ta pristop določene prednosti pred ostalimi.

3.2. Ostali pristopi k razvoju informacijskega sistema

Pri razvoju informacijskega sistema srečamo tudi druge metode razvoja, ki so se uveljavile v praksi. Posamezni avtorji predlagajo različne metode, zato naj predstavim še nekaj najpomembnejših:

- linearni model (Kovačič et al., 1994, str. 44),
- metoda prototipa (Kovačič et al., 1994, str. 44),
- razvoj s strani končnih uporabnikov (Gradišar et al., 2001, str. 435),
- uporaba programskih paketov (Gradišar et al., 2001, str. 436).

Linearni model lahko primerjamo z metodo življenjskega cikla, saj je ravno tako sestavljena iz zaporednih faz. Nobena od faz se ne more začeti pred zaključkom predhodne faze. Na koncu vsake faze se izdela poročilo, ki služi kot osnova za naslednjo fazo. Lastnosti linearnega modela lahko zapišemo z naslednjimi trditvami. Razvoj informacijskega sistema poteka zaporedno, skozi vnaprej določene faze. Vsaka faza je skrbno definirana in dokumentirana. Linearni pristop je bil prilagojen delu v velikih informacijskih centrih. Omogoča pregled nad posameznim projektom, standardizacijo postopkov in nadzor nad razvijalci informacijskega sistema. Poleg prednosti, pa ima linearni pristop tudi nekatere slabosti. V praksi je izredno težko določiti posamezne faze in zato prehodi med fazami niso jasni. Pogosto se je potrebno vračati nazaj, saj določena faza ni bila opravljena dovolj kvalitetno. Posledica so daljši

razvojni cikli, višji razvojni stroški in pomanjkljivosti se odkrije šele proti koncu projekta (Kovačič et al., 1994, str. 44-45).

Prototipna metoda razvoja informacijskega sistema se od klasične metode življenjskega cikla močno razlikuje. Pri prototipnem pristopu imamo rešitev že takoj na začetku v prvi fazi razvoja. Običajno prototip ni namenjen končni uporabi, temveč se z njegovo pomočjo izdela končni informacijski sistem. S preprostim prototipom lahko uporabnikom ali vodstvu prikažemo osnovne funkcije sistema, vhode, izhode in pri tem zanemarimo podrobnosti. Metodo prototipa so v praksi začeli uporabljati sorazmerno pozno, saj programska orodja niso omogočala prototipnega razvoja. Prototipni pristop ima določene prednosti. Skrajševanje časa potrebnega, da pridemo do prvih rezultatov in s tem nižji stroški razvoja. Pristop omogoča uporabnikom interaktivno sodelovanje z razvijalci sistema skozi celotno obdobje razvoja. S tem dosežemo, da napake in slabosti odpravimo v začetnih fazah razvoja. V praksi smo pogosto priča različnim pogledom na razvoj sistema. Metoda prototipa zelo dobro rešuje problem komunikacije, tako se verjetnost neprimernega končnega rezultata zmanjša. Prototipni pristop zahteva skrbno načrtovanje informatike na strateški ravni in podrobno opredelitev lastnosti na logični ravni (Kovačič et al., 1994, str. 47-48).

Z razvojem programske opreme, ki je bolj prijazna uporabnikom, se je razvil pristop imenovan razvoj informacijskega sistema s strani uporabnikov. V tem primeru končni uporabnik razvija sistem za lastno uporabo ob tehnični pomoči informatikov. Uporabniki morajo obvladati določen nivo znanja s področja informatike. Uporabniki se odločijo za takšen pristop predvsem zaradi boljšega poznavanja problematike. Tako zadostijo svojim specifičnim problemom, ki z vidika celotnega podjetja niso obravnavana. Kot rezultat razvoja končni uporabniki pridobijo znanja, za katere so morali včasih prositi programerje in informatike. Prednosti, ki jih prinaša pristop, so v zmanjševanju obremenjenosti informatikov, zato se lahko posvečajo zahtevnejšim projektom. Uporabniki ponavadi sami skrbijo za vzdrževanje svojih programskih rešitev. Kot avtorji rešitev, so do rešitev veliko bolj prizanesljivi ob napakah ali pomanjkljivostih. Velika slabost takih rešitev je slabša struktura in nesposobnost integriranja v celotni informacijski sistem (Gradišar et al., 2001, str. 435).

Pri vseh teh pristopih razvoja informacijskega sistema sodelujejo zaposleni v podjetju. Pri tem srečamo določene prednosti. Podjetja se pogosto odločajo za lastnem razvoj, ker trenutno na trgu ni razvite rešitve za njihovo problematiko. Domači informatiki najbolje zajamejo specifične zahteve podjetja. Nekateri programski paketi zahtevajo spremembe v poslovanju in organiziranosti podjetja. Ravno zato se v podjetjih odločijo, da bodo sami razvili informacijski sistem. S samostojnim razvojem se rešitve lažje prilagajo omejitvam strojne opreme in obstoječim informacijskim rešitvam (Gradišar et al., 2001, str. 436).

Naslednji način razvoja informacijskega sistema je nakup programskih paketov. Za posamezne tipe organizacij lahko dobimo že razvite informacijske sisteme, posebej na operativnem nivoju. Programski paketi so običajno sestavljeni iz modulov, ki pokrivajo določena področja delovanja podjetja. Nakup paketa uporabnikom skrajša čas uvedbe novega sistema. Zaradi množice že razvitih sistemov, moramo pri nakupu upoštevati paket z vsebinskega in tehničnega vidika, boniteto prodajalca, ceno paketa in ceno svetovanja in vzdrževanja. Prednosti nakupa paketov je v tem, da jih naredijo strokovnjaki s posameznega področja in so zato ponavadi boljši. V literaturi lahko zasledimo tudi nižje stroške razvoja, manjše število tehničnega osebja v podjetju in za bodoče spremembe bodo poskrbeli razvijalci sistema. Največja slabost je v specifičnih lastnostih posamezne organizacije. Kljub velikemu številu nastavitev je prostor prilagoditve sistema sorazmerno omejen. Ravno zaradi specifičnih lastnosti je potrebno sistem prilagoditi zahtevam podjetja. Nekatere prednosti, nakupa že izdelane rešitve, se s prilagajanjem izgubijo. V primeru, da dobavitelj prilagaja sistem, bo cena paketa višja, razvijalni čas daljši in na koncu bo sistem manj zanesljiv. Čeprav dobavitelji stalno posodabljajo obstoječe sisteme, ne bodo nudili posodobitev za prilagojene informacijske sisteme. V praksi smo priča uporabi vedno večjemu številu že razvitih programskih rešitev (Shelly et al., 1998, poglavje 5, str. 5).

3.3. Internet, Intranet, Ekstranet

Internet je javno in globalno komunikacijsko omrežje, ki zagotavlja neposredno povezavo med uporabniki s pomočjo lokalnega omrežja (LAN angl. local area network) ali ponudnika internetnih storitev. Internet je omrežje, ki se povezuje in usmerja preko usmerjevalnikov. Končni uporabniki so povezani na lokalne ponudnike internetnih storitev, ki so povezani na internetne ponudnike, do mrežnih ponudnikov in končno do internetne hrbtenice. Ker je Internet dostopen vsem, je opazno pomanjkanje nadzora, kar je vidno v velikem številu neurejenih informacij (Turban et al., 2000, str. 241-242).

Internet podpira programske rešitve v naslednjih večjih kategorijah (Turban et al., 1999, str. 169):

- Raziskovanje – vsebuje brskanje za informacijami. Uporabnikom omogoči pogled v dokumente.
- Komunikacija – internet predstavlja hiter in sorazmerno poceni komunikacijski kanal. Sporočila so lahko izredno kratka ali kompleksna poročila med podjetji. Vključuje tudi pretok informacij, elektronsko pošto in pogovor med skupinami.
- Sodelovanje – S pomočjo tehnologije lahko poteka sodelovanje med posameznimi skupinami. Najbolj znani načini sodelovanja so telekonference in ekranske konference.

Intranet je lokalno omrežje, ki uporablja internetno tehnologijo in je varovano s požarnim zidom. Požarni zid preprečuje vdor v sistem pred nepooblaščenimi uporabniki. Čeprav je intranet razvit na istem protokolu TPC/IP kot je Internet, deluje kot privatna mreža z omejenim dostopom. Ker intranet omogoča dostop skozi Internet, ne zahteva nobenega dodatnega omrežja. Ta odprta in fleksibilna povezava je glavna prednost uporabe intraneta v sodobnih organizacijah. Intranet se uporablja za poslovne programske rešitve znotraj podjetja.

Uporaba intraneta se zelo hitro širi v poslovnem svetu. Kalakota in Whinston (1997) opredelita intranet z naslednjimi funkcijami (Turban et al., 1999, str. 245-246):

- Učinkovit vnos transakcij – posamezni podatki so vneseni samo enkrat in se jih po potrebi posodobi. S takšnim vnosom se zmanjša napake.
- Enotno transakcijsko procesiranje – uporaba enotnih procesov v sistemu, kar zagotovi enotnost poročil.
- Stalna notranja kontrola – notranja kontrola nad vnesenimi podatki, procesi in poročili so standardizirani za celotno podjetje.

Ekstranet ali razširjeni intranet uporablja TPC/IP mrežni protokol Interneta, za povezavo intraneta na različne lokacije. Za transakcije, ki potekajo znotraj ekstraneta, moramo zagotoviti zasebnost in varnost. Večjo varnost zagotovimo s pomočjo predorov (angl. tunnels) zavarovanih tokov podatkov, s kriptografijo in avtorizacijskimi algoritmi. Ekstranet tako predstavlja povezavo med podjetniškim intranetom in ostalimi poslovnimi strankami. Dostop do intraneta je ponavadi omejen s dovolilnico, ki je skrbno varovana in omejena na določen krog ljudi. Z uporabo zaščitenega okolja ekstraneta omogočimo različnim uporabnikom sodelovanje, izmenjavo podatkov na varen način.

Tabela 1. Značilnosti interneta, ekstraneta in intraneta

Mrežni tip	**Uporabniki**	**Dostop**	**Vrste informacij**
Internet	Vsak posameznik z modemskim dostopom ali LAN	Neomejen	Splošne, javne
Intranet	Pooblaščeni uporabniki	Zaseben in omejen	Specifične, podjetniške
Ekstranet	Pooblaščene skupine poslovnih partnerjev	Zaseben in pooblaščeni zunanji partnerji	

Vir: Turban et al., 2000, str. 241.

Pri razvoju prvih internetnih informacijskih sistemov večina podjetij ni uporabljala metode življenjskega cikla. S temi projekti so si razvijalci šele pridobivali izkušnje z novo tehnologijo.

Podjetja so si sedaj že nabrala znanje in sedaj je čas, da začnejo z razvojem bolj formalnih informacijskih sistemov. Prvo vprašanje, na katerega moramo odgovoriti, je povezano z organiziranostjo spletne strani. Te se morajo skladati z organizacijsko strategijo. Pri izbiri organiziranosti spletnih strani lahko izbiramo med internetom, ekstranetom in intranetom. V prvi fazi razvoja moramo poskrbeti tudi za infrastrukturne zahteve in predvsem varnost. V organizacijah potrebujejo usposobljene zaposlene, ki lahko vzdržujejo spletne strani ali pa jim strokovno pomoč nudijo zunanji uslužbenci.

3.4. Varnost informacijskega sistema

Informacijski sistem je zgrajen iz mnogih komponent, ki so ponavadi na različnih lokacijah. Zato je vsak informacijski sistem ranljiv na več potencialnih lokacijah. Ranljivost informacijskega sistema se povečuje, če se podjetje poveže v svetovno mrežo. Teoretično je na stotine možnosti, ki omogočijo potencialno nevarnost za informacijski sistem.

Zaščita informacijske tehnologije je dosežena z dodajanjem zaščitnih mehanizmov, ki preprečijo možne nevarnosti, ugotovijo probleme čim bolj zgodaj, pospešijo popravljanje škode in popravijo problem. Nadzor je lahko vnesen v programsko opremo med fazo sistemskega razvoja. Dodatni sistemi za nadzor se lahko vnesejo tudi kasneje, ko je sistem že vzpostavljen ali med rednim vzdrževanjem.

Kontrolo informacijskega sistema lahko razdelimo v dve veliki skupini. Sistemska kontrola je vzpostavljena zato, da varuje sistem neodvisno od programskih rešitev. Na primer, zaščita strojne opreme in kontroliranje vnosa do podatkovnega centra sta neodvisna od posameznih programov. Aplikacijska zaščita je namenjena varnosti posameznim informacijskim sistemom.

Kategorije v sistemski kontroli so (Turban et al., 1999, str. 667-671):

- Fizična kontrola se navezuje na varnost računalniških kapacitet in virov. Fizična kontrola je ponavadi prva linija kontrole in jo je tudi najlažje zagotoviti. Varuje sistem pred naravnimi nesrečami in človeškimi napakami.
- Dostopna kontrola je omejena na nedovoljen vstop v informacijski sistem. Uporabniki morajo za vstop dobiti avtorizacijo. Dostop do računalniškega sistema je sestavljen iz treh faz, fizični dostop do terminalov, dostop do sistema in dostop do specifičnih transakcij, pravic, programov in podatkov znotraj sistema. Programska oprema za dostopne kontrole je na voljo velikim računalnikom, osebnim računalnikom, lokalnim mrežam in omrežju. Najpogostejše oblike dostopne kontrole so zagotavlja z:
  - gesli,
  - pametno kartico ali žetoni.

- V praksi se lahko uporablja eno od naslednjih biometričnih naprav:
  - **Slika**- računalnik slika vaš obraz in sliko primerja s sliko v računalniku.
  - **Prstni odtis**- vsakič ko želimo vstopiti v sistem moramo s prstnim odtisom pokazati istovetnost.
  - **Oblika roke**- računalniško sliko roke primerja s sliko v spominu.
  - **Glas**- računalnik primerja glas.
  - **Dinamika pritiskanja tipkovnice**- računalnik primerja hitrost in moč pritiska na tipkovnico.
  - **Podpis**- primerja prejšnji podpis.
- Varnost podatkov pred namernim posegom v sistem s strani zunanjih oseb. Varnost podatkov zagotovimo skozi operacijski sistem s programi za varen dostop do podatkov in baze podatkov. V podjetju naj bi se držali dveh osnovnih načel. Uporabniku dodelimo samo tiste pravice, ki jih potrebuje. Ko uporabnik dobi pravico dostopa do pomembnih podatkov, je njegova odgovornost, da podatki ne pridejo v roke nepooblaščenih uporabnikov.
- Komunikacijske kontrole so postale izredno pomembne zaradi uporabe interneta, intraneta in elektronskega poslovanja.
- Administrativna kontrola se ukvarja z usmerjanjem in nadzorovanjem. Te kontrole vključujejo naslednje:
  - Primerna izbira, izobrazba in nadzor zaposlenih, posebej v računovodstvu in informatiki.
  - Zagotavljanje organizacijske lojalnosti.
  - Periodično menjavanje pravice dostopov, kot so gesla.
  - Razvijanje programskih standardov.

Sistemske kontrole varujejo računalniške naprave in zagotavljajo varnost fizični opremi, programski opremi, podatkom in mrežam. Sistemska kontrola ne more zagotoviti varnosti vsebine posameznih aplikacij. Zato so kontrole pogosto vgrajene v programske rešitve in so ponavadi zapisane kot pogoji. Lahko jih razdelimo v tri večje kategorije: vnosna kontrola, procesna kontrola in kontrola pri izhodu (Turban et al., 1999, str. 671-672).

- Vnosna kontrola je narejena, da zagotavlja pravilnost podatkov. Podatki morajo biti natančni, popolni in konsistentni. Njihova osnovna naloga je preprečevati smeti-noter smeti-ven. Kar pomeni, da je kakovost dobljenih informacij odvisna od kakovosti vhodnih podatkov (angl. GIGO garbage-in, garbage-out).
  - Popolnost- določeni podatki morajo biti izpolnjeni, npr.: naslov, občina, pošta in poštna številka.
  - Oblika- oblika je lahko standardizirana.
  - Velikost- nekateri podatki so lahko samo znotraj določenih meja.
  - Doslednost- podatki iz različnih virov morajo biti primerljivi.

- Procesna kontrola zagotavlja popolnost, pravilnost in natančnost podatkov v času programske obdelave podatkov.
- Kontrole pri izhodu zagotavljajo, da so izpisi pravilni, točni, popolni in konsistentni.

4. Načrtovanje informacijskega sistema

Vsako podjetje mora imeti izdelan plan za prihodnost, imenovan vizija podjetja. V viziji so zapisane podjetniške smernice za prihodnost in ponavadi vključuje glavne proizvode, storitve in vrednote. Pri definiciji vizije, si lahko podjetje postavi cilje, ki jih bo izpolnilo v prihodnjih letih. Pri razvoju informacijskega sistema mora analitik upoštevati vizijo podjetja in oblikovati takšen sistem, ki bo omogočal, da se bo podjetje razvijalo v skladu z vizijo podjetja (Shelly et al., 1998, poglavje 2, str. 3).

Golf zveza Slovenije se v zadnjem desetletju izredno hitro širi. Število golf igrišč in igralcev je v Sloveniji vsako leto večje in ena od nalog vsake zveze je poskrbeti za razvoj športa. Z večjo množičnostjo so si v Golf zvezi Slovenije postavili tudi zelo visoke cilje glede tekmovalnega golfa. Da bi te cilje lahko uresničili potrebujejo tudi informacijski sistem, ki bo pokrival analizo treningov in turnirjev vrhunskih igralcev. Do sedaj igralci niso uporabljali računalniško podprte analize. Večina statistike in analiz je bila opravljenih v papirni obliki, pa še te so izdelovali le posamezniki. Trenerji in odgovorni teh podatkov niso imeli, tako da je bilo njihovo odločanje veliko težje. Na pobudo tekmovalnega odbora pri golf zvezi Slovenije, ki je hkrati tudi naročnik informacijskega sistema, smo se odgovorili, da začnem razvijati sistem, ki bo pokrival potrebe po analizi treningov in turnirjev.

4.1. Razlogi za uvedbo IS

V literaturi lahko zasledimo, da je začeten povod za razvoj informacijskega sistema sistemska zahteva (angl. systems request). V zahtevi so podani vzroki za spremembe in ti so lahko izboljšava obstoječega sistema, odprava napak ali razvoj novega informacijskega sistema. Golf zveza Slovenije trenutno ne uporablja nobenih analiz treningov in turnirjev, zato bo potrebno razviti informacijski sistem od začetka (Shelly et al., 1998, poglavje 2, str. 7-8).

V konkretnem primeru so glavni vzroki za razvoj informacijskega sistema naslednji:

- Izboljšano delovanje – z novim informacijskim sistemom bodo trenerji in učitelji golfa bolj pregledno vodili tekmovalni šport. Sponzorji bodo lahko na spletni strani videli določene informacije o treningu. Z novimi informacijami bodo priskrbljeni tudi uporabniki sistema, saj trenutno nihče ne izdeluje tako popolne statistike.

- Več informacij – trenerji in profesionalni učitelji golfa bodo od svojih igralcev dobili povratno informacijo. Na podlagi statistik bodo prilagajali trening za posamezne igralce. Z analizo treninga in rezultatov se bo lahko ugotovilo učinkovitost treningov in v kateri smeri morajo tekmovalci še trenirati.
- Zmanjšanje stroškov – vsi vemo, da so ure učenja golfa izredno drage in trening, ki ne da pravih rezultatov, je izredno drag. Z novimi informacijami, bi se stroški treninga znižali, saj bi tekmovalci trenirali prave stvari. Na ta način se bo povečala uspešnost igralcev golfa.

Ideje za razvoj informacijskega sistema lahko pridejo od notranjih in zunanjih udeležencev. Med notranje udeležence sodijo obstoječi sistem, vodstvo in uporabniki. Največkrat so ravno uporabniki tisti, ki predlagajo nove ideje, saj so ravno oni neposredno povezani z informacijskim sistemom. Tudi v našem primeru lahko rečem, da so prve zahteve oziroma ideje prišle s strani igralcev. Ravno oni so najbolje seznanjeni z informacijskimi potrebami. Zaradi tega dejstva bodo ravno ti igralci tudi največji vir informacijskih zahtev. Vodstvo Zveze je idejo o novem informacijskem sistemu podprlo in bo nudilo pomoč pri izdelavi projekta. Med zunanjimi faktorji lahko izpostavim konkurenco v smislu tekmovalnosti. Ravno vedno večja tekmovalnost v športu je spodbudila razmišljanje o uporabi računalnika pri analizi treninga in turnirjev.

4.2. Izvedljivost IS

V večini podjetij dobi informacijski oddelek več sistemskih zahtev, kot jih lahko podjetje odobri. Odbor analizira posamezne zahteve in izbere najbolj primerne projekte. V vsaki organizaciji imajo omejena sredstva in zaposlene, zato morajo postaviti prioritete pri izbiri sistemskih zahtev.

Sistemsko zahtevo moramo pred začetkom testirati in ugotoviti, ali je vredno projekt nadaljevati ali ne. Ta skupek testov se imenuje študija izvedljivosti (angl. feasibility study) in je pomemben del vsakega razvoja informacijskega sistema. Študija izvedljivosti uporablja tri glavne sklope izvedljivosti, to so operativna izvedljivost, tehnična izvedljivost in ekonomska izvedljivost (Shelly et al., 1998, poglavje 2, str. 13).

Prvi korak prvi ugotavljanju izvedljivosti je postavitev ključnih vprašanj. Vsako zahtevo, ki ni izvedljiva, je potrebno čim preje izločiti. Včasih so projekti tehnično izvedljivi, vendar se nam njihov razvoj vseeno ne izplača. Pri tem moramo imeti pred očmi, da nekateri sistemi niso izvedljivi trenutno, vendar bodo v bližnji prihodnosti s prihodom nove programske ali strojne opreme, stroški razvoja se bodo znižali ali koristi toliko narasle. Včasih se zgodi, da

popolnoma izvedljiv projekt kasneje med samim razvojem zavržemo. Lahko se spremenijo ekonomske razmere, ali pa postanejo stroški večji kot smo pričakovali. Tudi končni uporabniki ali vodstvo lahko izgubi zaupanje v sistem in to pomeni njegov konec. Zaradi vseh teh razlogov je študijo izvedljivosti potrebno opravljati v posamezni fazi v modelu življenjskega cikla (Shelly et al., 1998, poglavje 2 str.13).

Operativna izvedljivost (angl. operational feasibility) nam pove ali bo informacijski sistem dosegel svoj namen. V primeru, da bodo uporabniki imeli težave pri uporabi sistema, verjetno nismo dosegli svojega namena. Operativna izvedljivost je odvisna od mnogih dejavnikov (Shelly et al., 1998, poglavje 2, str. 14).

V konkretnem primeru so ključna vprašanja o operativni izvedljivosti naslednja:

Ali uporabniki in vodstvo podpirata razvoj novega projekta?

- Razvoj informacijskega sistema so najprej podprli uporabniki. Programska rešitev jim bo olajšala delo. Tudi v vodstvu so projekt sprejeli pozitivno in so pripravljeni sodelovati pri razvoju.

Ali bodo uporabniki resnično vnašali podatke?

- Večina igralcev že sedaj vodi statistiko, vendar v papirni obliki. Z uporabo novega sistema bo tako čas, potreben za vpisovanje podatkov krajši. Hkrati bodo igralci dobili več informacij o svoji igri. Da bodo igralci resnično vnašali podatke, bo skrbel trener, ki bi na podlagi izpisov oblikoval količino treninga.

Ali bo potrebno uporabnike ponovno usposobiti in kako bomo to izvedli?

- Sistem bo zasnovan tako, da bo čim bolj enostaven za uporabo. Od uporabnikov se bo zahtevalo le osnovno predznanje uporabe računalnika. Predstavitev programske rešitve bo uporabnikom, predstavljena v reviji Golf. V članku bo pojasnjeno osnovno delovanje z navodili za uporabo.

Ali bodo uporabniki vključeni v planiranje?

- Uporabniki bodo sodelovali v začetnih fazah. Sistem bo oblikovan tako, da bo zadostil njihovim potrebam. Uporabnost informacijskega sistema bo odvisna od pokrivanja resničnih potreb.

Na vsa ta vprašanja moramo odgovoriti pozitivno, da lahko rečemo, da je projekt operativno izvedljiv.

Ko projekt prestane operativno izvedljivost, lahko začnemo s tehnično izvedljivostjo (angl. technical feasibility). Sistem je tehnično izvedljiv v primeru, da ima organizacija vsa potrebna

sredstva za razvoj ali nakup, namestitev in izvajanje sistema (Shelly et al., 1998, poglavje 2, str. 14). Pri tehnični izvedljivosti moramo biti pozorni na naslednje točke.

Ali ima organizacija potrebno opremo?

- Za razvoj informacijskega sistema bo uporabljena obstoječa programska in strojna oprema. Uporabniki imajo dostop do osebnih računalnikov. Golf zveza Slovenije trenutno nima dostopa do spletnega strežnika. V primeru internetne programske rešitve, bi dostop do strežnika lahko zagotovili.

Ali ima oprema dovolj kapacitet za bodočo uporabo?

- Kapaciteta diskov bo za uporabo zadostovala. Zmogljivosti bodo zadostne tudi v primeru širitve informacijskega sistema v prihodnosti.

Ali se bi sistem lahko vključil v že obstoječi informacijski sistem?

- Trenutno pri Zvezi ne uporabljajo celovitega informacijskega sistema.

Ali je nujna internetna tehnologija?

- V začetni fazi, bi lahko programsko rešitev uporabljali tudi na posameznih osebnih računalnikih. Informacijski sistem mora biti zasnovan tako, da dopusti prehod na internetno tehnologijo.

Ali imajo razvijalci zadostna znanja za razvoj končne programske rešitve?

- Za razvoj informacijskega sistema bom zadolžen sam. Pri načrtovanju in razvoju informacijskega sistema se lahko obrnem na profesorje na fakulteti, ki mi bodo nudili pomoč.

Zadnja vrsta izvedljivosti se imenuje ekonomska izvedljivost (angl. economic feasibility). Sistem je ekonomsko izvedljiv, če so koristi, večje kot so stroški razvoja. Stroški so lahko enkratni ali pa nastajajo skozi celotno obdobje uporabe sistema. Pri stroških moramo upoštevati področja kot so strojna oprema, programska oprema, razvoj ali nakup, izobraževanje uporabnikov, licence, stroške konzultantov in še druge stroške. Na drugi strani je potrebno oceniti koristi, ki so lahko otipljive ali neotipljive vendar imajo vseeno pomemben vpliv na organizacijo. Na koncu je potrebno tudi razmisliti o stroških, ki jih imamo, če informacijskega sistema ne razvijemo (Shelly et al., 1998, poglavje 2 str.15).

Dve ključni vprašanji v primeru ekonomske izvedljivosti v primeru razvoja novega informacijskega sistema za Golf zvezo Slovenije sta:

Kakšni so stroški razvoja za Golf zvezo Slovenije?

- Načrtovanje in razvoj bom izvedel samostojno in bo brezplačno. Programsko orodje za razvoj programske rešitve Visual Basic imam že nameščen na osebnem računalniku, tako da dodatnih stroškov ni bilo. Hkrati jim bom lahko nudil tudi pomoč pri usposabljanju uporabnikov. Stroški, ki bi nastali, bi bili povezani z najemom strežnika za internetno programsko rešitev. Strošek, ki ga je težko ovrednotiti, bo čas, ki ga bodo uporabniki potrebovali za uvajanje. Uporaba bo preprosta, zato prehod na nov sistem ne bo težak.

Koristi zaradi razvoja programske rešitve?

- Koristi zelo težko opredelimo v denarju. Največje koristi bodo imeli igralci in trenerji, saj bodo dobili zelo dobre povratne informacije o preteklem delu.

Glede na ta dejstva lahko rečem, da je študija izvedljivosti pokazala, da je projekt izvedljiv in da bodo njegove koristi večje od stroškov.

5. Analiza informacijskega sistema

V drugi fazi metode življenjskega cikla je naloga analitika zbrati informacije o obstoječem sistemu, lahko je računalniški ali papirni, in raziskati potrebe novega informacijskega sistema. Sistem zbiranja informacij pomaga analitiku in managerju spoznati potencialne spremembe v obstoječem sistemu. Prva faza je določitev in zbiranje zahtev. V drugi fazi analitik analizira zahteve uporabnikov in vodstva. V zadnji fazi analitik predstavi možne alternative in poda poročilo naročniku projekta.

Analiza informacijskega sistema je lahko izredno težka v primerih velikih sistemov. Osrednja naloga sistemskega analitika je poznavanje dinamike informacijskega sistema in poslovnega procesa. Informacijski sistemi niso statični, zato se morajo hitro prilagajati zunanjim spremembam okolja. Za uspešno analizo sistema je potrebno kritično razmišljanje in dobri medosebni odnosi (Shelly et al., 1998, poglavje 3, str. 5).

Sistemski analitik potrebuje jasne odgovore na pet vprašanj. V naslednjih točkah je opisan proces izvajanja treningov in turnirjev.

Kdo? Kdo izvaja posamezne operacije? Ali jih izvajajo lahko drugi ljudje?

- Glavni udeleženci v procesu so igralci sami. Če želijo igralci igrati na turnirjih, se je potrebno prej prijaviti. Turnirje organizirajo posamezna golf igrišča, ali pa tudi golf zveze in klubi. Podatke o igri, si igralci med igro zapisujejo na števne kartice. Na turnirjih si igralci izmenjajo kartice, tako da vsak piše za drugega igralca.

Kaj? Kaj se v procesu dogaja? Kako si procesi sledijo? Zakaj je proces potreben?

- Igralci opravljajo treninge na vadišču in igrišču v okviru planov, ki jim jih postavi trener. Trening je razdeljen na več delov igre, od patanja, kratke igre do udarcev z lesovi. Poleg treninga na vadišču, pa igralci opravljajo tudi treninge na igrišču.

Kje? Kje se operacije izvajajo? Zakaj? Kje bi se lahko izvajale? Ali bi jih lahko bolj učinkovito izvajali drugje?

- Igra na turnirjih vedno poteka na zunanjih golf igriščih. Turnirji potekajo tako v Sloveniji, kot tudi v tujini. V tujini se igralci udeležujejo predvsem mednarodnih prvenstev. Trening ponavadi poteka na domačih igriščih posameznih igralcev. Poleg treninga golfa, pa se vedno več igralcev udeležuje kondicijskih treningov. Ti se izvajajo na prostem, predvsem tek in kolesarjenje, in v telovadnicah ali fitness centrih. Podatki o igri se beležijo na posebno kartico, ki je osnova za računanje rezultatov. Tudi v primeru informatizacije, bodo igralci morali še vedno uporabljati števno kartico, saj je to edini uradni dokaz o igralčevem rezultatu.

Kdaj? Kdaj se operacije izvajajo? Zakaj ravno ta čas?

- Največ turnirjev je organiziranih v obdobju od aprila do oktobra. V tem obdobju je kvaliteta golf igrišč v Evropi tudi najboljša. Z zimskem času je večina igrišč zaprtih, zato poteka tudi manj tekmovanj. Turnirji lahko potekajo en dan ali več, odvisno od vrste tekmovanj. Treningi potekajo čez vse leto. V poletnih mesecih je večji poudarek na igranju na igrišču. V zimskih mesecih pa se količina treninga zmanjšuje in je poudarek na kondiciji.

Kako? Kako je operacija opravljena? Zakaj je opravljena na ta način?

- Turnirji potekajo po različnih sistemih igre. Najbolj običajna sta seštevna igra in stableford. Skupni zmagovalec je tisti, ki doseže skupno najmanjše število udarcev v vseh turnirskih dnevih. Vsak igralec dobi na začetku igre števno kartico, na katero med samo igro zapisuje svoje udarce oziroma v primeru turnirja, od igralca v svoji skupini. S podpisom na kartici, igralci jamčijo, da so podatki pravilni.

5.1. Določitev zahtev

Sistemske zahteve so značilnosti, ki naj bi jih vseboval bodoči informacijski sistem. Prepoznavanje potreb je ključno, saj nam le te dajo osnovo za razvoj informacijskega sistema. Te zahteve nam bodo služile tudi kot kontrola ali smo izdelali informacijski sistem v skladu s poslovnimi zahtevami.

Sistemske zahteve lahko razdelimo v pet kategorij: vhodi, izhodi, procesi, usklajenost in kontrola. Pri določanju sistemskih zahtev so pomembne tudi informacije o sedanjem in bodočem obsegu, količini in pogostosti transakcij. Vsi ti podatki nam pomagajo pri izbiri prave strojne in programske opreme (Shelly et al., 1998, poglavje 3 str. 5-6).

Naslednji korak se imenuje zbiranje zahtev (angl. fact-finding), ki vsebuje različne tehnike kot so intervjuji, pregledi dokumentacije, opazovanje, anketiranje in raziskovanje. Na tem mestu sistemski analitik porabi veliko časa za pogovore z bodočimi in sedanjimi uporabniki (Shelly et al., 1998, poglavje 3 str. 7).

Sam sem zbiral podatke z različnimi metodami. Zbiranje sistemskih zahtev je zelo zapleteno, saj mora analitik programske zahteve prevesti v jezik uporabnika. Velikokrat uporabnik tudi točno ne ve, kaj kakšen parameter pomeni in kaj točno bo programska rešitev delala. Ker tudi sam igram golf, se dobro spoznam na zahteve in želje uporabnikov. Vseeno sem se odločil, da dobim informacije tudi od drugih igralcev golfa. Izvedel sem anketo v kateri je sodelovalo 53 igralcev in igralk golfa iz Slovenije, med katerimi so bili tako amaterski kot tudi profesionalni igralci. V anketo so bile vključene moška, ženska in mladinska državna reprezentanca Slovenije in še nekateri drugi igralci golfa. Anketa je v prilogi številka 1. Poleg ankete sem določene zahteve dobil tudi z razgovori, ki sem jih imel s posameznimi igralci. Določene ideje o najpomembnejših izhodnih elementih informacijskega sistema sem dobil tudi na internetnih straneh, ki vodijo turnirske statistike za najboljše igralce na svetu. Podrobnejše analize zahtev sledijo v naslednjem poglavju.

5.2. Analiza sedanjega stanja

Naslednja faza v metodi življenjskega cikla je analiza pridobljenih zahtev. Na tem mestu bom predstavil dobljene rezultate ankete, razgovorov in analizo internetnih strani, ki vodijo statistiko.

Anketa je bila sestavljena iz 16 vprašanj. Vsa vprašanja so imela že predvidene odgovore, čeprav je bila pri nekaterih dopuščena možnost, da sami dodajo svojo rešitev. Na začetku so bila splošna vprašanja, nato pa še vprašanja v zvezi s programsko rešitvijo. Na prvo vprašanje, če imajo osebni računalnik, je 95 odstotkov odgovorilo pozitivno. Deset odstotkov anketirancev že uporablja računalniško podprt sistem za vodenje svojega treninga in turnirjev. Med vsemi sta samo dva, ki sta programsko rešitev kupila, vsi ostali uporabljajo programsko rešitev, ki je na voljo na internet strani brezplačno. Po drugi strani pa bi morebitni informacijski sistem uporabljalo kar 85% vseh vprašanih. Ker sem sam izbiral anketirance, je odstotek precej višji, kot bo kasneje v praksi. Kriterij za izbiro anketirancev je bila kvaliteta

igralcev. Pri anketi sem želel dobite predvsem potrebe najboljših igralcev in hkrati tistih, ki bodo programsko rešitev kasneje tudi uporabljali. Ocenjujem, da bo nov informacijski sistem uporabljalo okoli sto igralcev, to pa so tisti, ki igrajo golf tekmovalno. Vsi anketiranci uporabljajo operacijski sistem Windows.

Naslednji sklop vprašanj se je navezoval na zahteve sistema. Zanimalo me je tudi, koliko časa so anketiranci pripravljeni nameniti vnosu podatkov. S tem vprašanjem sem želel dobiti informacijo, kako obsežna naj bo programska rešitev. 45 odstotkov je izbralo največjo postavljeno možnost to je od šest do deset minut, 30 med tri in pet minut in ostali eno do dve minuti. Glede na te odgovore lahko pripravim informacijski sistem, ki bo zahtevala vnos okoli pet minut. Z vprašanjem številka sedem, sem dobil zahteve po izračunih. Najpogostejše zahteve so bile število udarcev po luknjah, število puttov po luknjah, število zadetih zelenic, hendikep, število zadetih čistin. Nekoliko manj anketirancev bi poleg teh kazalcev potrebovalo še letno povprečno število puttov na krog, letno povprečno število udarcev, letni povprečni odstotek zadetih zelenic in čistin, število sand saves na krog in število up and down na krog. Poleg teh kazalnikov pa so nekateri igralci dodali še svoje zahteve kot so statistika treninga, posamezne kazalnike lahko izrazimo tudi v odstotkih. Nekateri so želeli imeti tudi prostor za opis izvedbe posameznega udarca in težav na igrišču. Nekaj več kot 50 odstotkov bi poleg turnirjev in treninga na igrišču zapisovalo tudi trening na vadišču.

Na vprašanji o izpisih in pogostosti izpisov je večina odgovorila, da bi želele imeti podatke predstavljene v obliki tabele in grafa. Izpise bi potrebovali mesečno, nekoliko manj dnevno in letno. Poleg teh osnovnih možnosti pa so bile podane zahteve po turnirskem in tedenskem izpisu.

Naslednji sklop vprašanj se je navezoval na različna okolja. Predvsem me je zanimalo, ali naj programsko rešitev pripravim za osebni računalnik, internetno okolje ali dlančnik. 95 odstotkov vprašanih je odgovorilo, da ima povezavo z Internetom in 80 % bi uporabljalo programsko rešitev, če bi bila v internetnem okolju. Na vprašanja v zvezi z dlančnikom, je 85 odstotkov odgovorilo, da ga trenutno nima. Približno polovica anketirancev bi podatke vnašala preko dlančnika, čeprav bi jih le 20 odstotkov vnašalo že med samo igro. Na zadnje vprašanje, o najljubšem okolju, je 45 odstotkov odgovorilo z uporabo internetne rešitve, 45 odstotkov z uporabo informacijski sistem na osebnem računalniku in le deset odstotkov je za najprimernejši način izbralo vnos preko dlančnika.

Na internetnih straneh je možno videti različne statistike za profesionalne igralce. Njihova statistka je izredno podrobna, vendar pa se moramo zavedati, da samim igralcem ni potrebno vnašati podatkov, saj to za njih opravljajo drugi. Ideje, ki bi lahko bile dobre tudi za našo programsko rešitev, so odstotek zadetih zelenic na parih 3, odstotek zadetih zelenic na parih 4,

odstotek zadetih zelenic na parih 5, povprečen rezultat na parih 3, povprečen rezultat na parih 4 in povprečen rezultat na parih 5.

5.3. Ocena možnih alternativ

S pomočjo matrike prednosti in slabosti bom ocenil posamezno možnost razvoja programske rešitve. Na podlagi dejstev in tudi analize zahtev bom izbral najboljšo rešitev in to rešitev razvijal naprej.

Možne so tri alternative. Prva je klasičen vnos preko osebnega računalnika. Uporabniki imajo na svojem osebnem računalniku programsko rešitev, ki jo lahko uporabljajo samo na tem računalniku. Drugi način je vnos preko interneta, kjer so podatki shranjeni vsi v eni bazi podatkov. Dostop je tako možen na vseh računalnikih, ki imajo povezavo z internetom. Tretji način dostopa do podatkov je preko dlančnika. Dostop je možen praktično v vsakem trenutku in na vsakem kraju.

V naslednjih tabelah so prikazane bistvene prednosti in slabosti posamezne programske rešitve.

Tabela 2. Prednosti in slabosti razvoja informacijskega sistema za osebni računalnik

Prednosti:	**Slabosti:**
• Osebni računalnik ima skoraj vsak igralec golfa. Anketa je pokazala, da samo 4 % vprašanih nima osebnega računalnika. • Večja varnost podatkov, saj ima vsak dostop samo do svoje baze. • Ni potrebno imeti velike skupne baze. • Ni potrebno posebej dodeljevati pravice uporabnikom. • Hitrost odzivnosti programske rešitve je večja. • Nižji stroški razvoja • Posamezna programska rešitev deluje neodvisno od ostalih.	• Vnos je možen samo na enem računalniku

Tabela 3. Prednosti in slabosti razvoja informacijskega sistema kot internetno rešitev

Prednosti:	**Slabosti:**
• Skoraj vsi vprašani, ki imajo osebni računalnik, imajo tudi dostop do interneta. • Vnos možen na vseh računalnikih z internetno povezavo. • Z dodeljevanjem pravic je možen vpogled v več uporabnikov hkrati.	• Golf zveza Slovenije ima dodatne stroške najema spletnega strežnika in baze podatkov. • Manjša varnost podatkov. • Ker večina igrišč ne nudi prostega dostopa do interneta, bi igralci vnašali podatke na domačem računalniku, tako da prednost vnosa na različnih lokacijah ni tako velika. • Počasnejše delovanje. • Uporabnik ima pri uporabi stroške povezave z internetom. • V primeru izpada strežnika, imajo vsi uporabniki onemogočen dostop do podatkov.

Tabela 4. Prednosti in slabosti razvoja informacijskega sistema za dlančnik

Prednosti:	**Slabosti:**
• Dostop do podatkov v realnem času. • Možnost razvoja programske rešitve v prihodnosti	• Zelo malo anketirancev že ima dlančnik, v anketi samo 15 % vprašanih. • Samo 20 % vprašanih bi vnašalo podatke o svoji igri že med samo igro na igrišču. Verjetno bi bila končna številka še manjša, saj je takojšnje vnašanje podatkov zamudno. • Uporaba električnih pripomočkov je na turnirjih prepovedana. • Zaradi novega okolja, je razvoj novih rešitev negotov

5.4. Izbor alternative

Na podlagi prednosti in slabosti posameznih alternativnih okoljih, je najboljša programsko rešitev, ki bo delovala samo na osebnem računalniku. Trenutno je dlančnik najslabša možnost. Največja slabost dlančnika je v zelo slabi razširjenosti med igralci. Pa tudi njegova edina velika prednost ne zdrži na turnirjih. Razvoj programske rešitve za dlančnik ostaja kot možnost razvoja v prihodnosti. Ostali dve alternativi sta si zelo izenačeni, tako po prednostih in slabostih, kot tudi med samimi anketiranci. Glede na dejstvo, da bo to prva programska rešitev in hkrati tudi poskusna, je smiselna odločitev za razvoj informacijskega sistema, ki bo deloval na osebnem računalniku. Tako programsko rešitev za dlančnik, kot tudi internetno rešitev je možno razviti v prihodnosti.

6. Zasnova informacijskega sistema

V tretji fazi življenjskega cikla razvoja začne sistemski analitik na podlagi logičnega modela, ki ga je razvil v drugi fazi, razvijati fizični model programske rešitve. Najprej mora izbrati in razviti bazo podatkov. Glede na dejstvo, da vsi anketiranci uporabljajo operacijski sistem Windows in da bodo baze podatkov razdrobljene po posameznih računalnikih, bom za bazo podatkov uporabil Microsoftovo orodje Access, ki deluje v okolju Windows. Access je relacijska baza podatkov in je primeren za manjše osebne baze podatkov, ki ne vsebujejo velike količine podatkov. Ko imamo izbrano bazo podatkov lahko naredimo relacijski model in istočasno tudi fizični model baze podatkov. V zaključku tretje faze analitik izdela vnosne forme bodočega informacijskega sistema in potencialne izpise. Na tem mestu lahko analitik sodeluje s končnimi uporabniki in vodstvom. Ko se naročniki razvoja sistema strinjajo z zasnovo informacijskega sistema, lahko programerji začnejo razvijati programsko rešitev.

6.1. Izgradnja baze podatkov

Za dobro delovanje končne informacijskega sistema je načrtovanje baze podatkov izredno pomembno. Čeprav ima sistemski analitik v tej fazi zelo malo prostora, saj je izgradnja baze podatkov bolj tehnično opravilo, pa še vedno sodeluje z uporabniki. Z dobro načrtovanjem podatkovne baze se kasneje izognemo težavam pri implementacij.

V procesu izgradnje baze podatkov je potrebno poznati organiziranost podatkov in strukturo podatkov, ki so potrebni za delovanje informacijskega sistema. Sistemski analitik na podlagi predhodnih faz lahko ugotovi povezanost med posameznimi podatki in tudi tip in velikost podatkov. Tip podatkov je izredno pomemben, saj lahko z določenimi kontrolami pri vnosu

zmanjšamo število nepravilnih vnosov. Dobro poznavanje velikosti podatkov pa nam služi za optimizacijo velikosti baze podatkov. V tej fazi moramo tudi razmisliti o kandidatih za ključ, ki ga bomo kasneje uporabili v relacijski bazi podatkov. Glavni ključ razlikuje entitete nekega tipa med seboj. V relacijski bazi podatkov, tako dve vrstici ne moreta imeti iste vrednosti glavnega ključa.

6.1.1. Model entitet-povezav

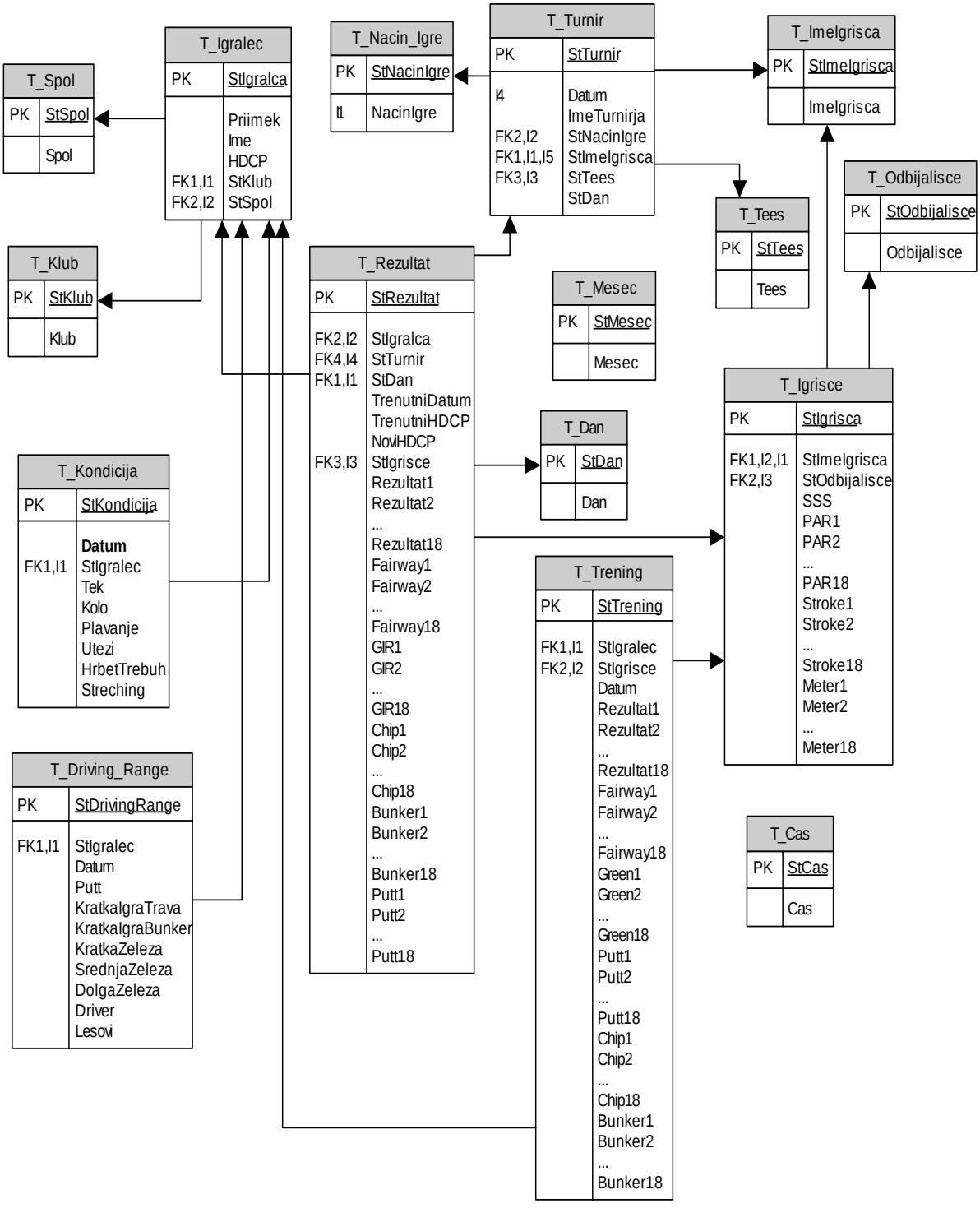
V diagramu entitet-povezav so prikazane posamezne entitete in njihove odvisnosti med seboj. Diagram entitet-povezav ne prikazuje toka podatkov, zato med entitetami puščice, ne prikazujejo smer gibanja podatkov. Pozicija posameznih entitet nam ne pokaže nadrejene oziroma podrejene zveze med entitetami (Shelly et al., 1998, poglavje 8, str. 5).

Med entitetami so možne tri vrste povezav in sicer 1:1 , 1:M ter M:N povezava. Bistvo baz podatkov je, da se podatki zapišejo v bazo samo na enem mestu. Normalizacija je proces s katerim odstranimo podvojene zapise. Ko imamo bazo podatkov normalizirano je spreminjanje podatkov enostavno, saj popravimo podatek samo na enem mestu in ta sprememba vpliva na vse zapise v bazi (Shelly et al., 1998, poglavje 8, str. 5-7).

Tabela 5: Vse tabele v bazi podatkov

Ime tabele	**Opis**
T_Igralec	Vsebuje vse podatke o igralcu
T_ImeIgrisca	Zapisani so vsa imena igrišč
T_Klub	Zapisani so golf klubi igralcev
T_Odbijalisce	Zapisana so odbijališča na igriščih
T_NacinIgre	Vsi možni načini igre
T_Rezultat	Vsi podatki o igri na turnirjih
T_Trening	Vsi podatki o igri na treningu
T_Igrisce	Podatki o posameznem igrišču
T_Turnir	Podatki o turnirjih
T_Kondicija	Podatki o kondicijskih treningih
T_Driving_Range	Podatki o treningu z vadišča
T_Spol	Spol igralcev
T_Dan	Trajanja turnirja
T_Tees	Podatki o odbijališčih
T_Cas	Časovne možnosti treninga

Slika 3: E-R diagram za končno programsko rešitev



6.1.2. Fizični model

Na podlagi diagrama entitet-povezav sistemski analitik sestavi fizični model baze podatkov. V fizičnem modelu so opisane posamezne spremenljivke. Določene so jim vrste podatkov, kot na primer tekstovni, numerični podatek ali datum. Za posamezne podatke je potrebno določiti še velikost in obliko podatkov. V primeru nejasnosti se lahko ponovno sestanemo z uporabniki in določimo velikosti podatkov. Z določanjem velikosti posameznih polj rezerviramo prostor v bazi za naše zapise. Na tem mestu lahko postavimo tudi obliko podatkov, predvsem datumov in posameznim šifram, ki jih uporabljajo v poslovnem procesu.

Tabela 6. Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Igralec

Ime polja	**Podatkovni tip**	**Opis**
StIgralec	Samoštevilo	Zaporedna številka igralca
Priimek	Besedilo	Priimke igralca
Ime	Besedilo	Ime igralca
HDCP	Število	Zaokroženo na eno decimalno mesto
StSpol	Število	Zaporedna številka spola
StKlub	Število	Zaporedna številka kluba

Tabela 7. Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Turnir

Ime polja	**Podatkovni tip**	**Opis**
StTurnir	Samoštevilo	Zaporedna številka turnirja
ImeTurnirja	Besedilo	Ime turnirja
StImeIgrisca	Število	Številka igrišča na katerem se turnir igra
StDan	Število	Število turnirskih dni
StOdbijalisca	Število	Številka odbijališč na turnirju
StNacinIgre	Število	Številka Načina igre na turnirju
Datum	Kratki datum	Datum turnirja

Tabela 8. Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Rezultat

Ime polja	**Podatkovni tip**	**Opis**
StRezultat	Samoštevilo	Zaporedno število rezultata
StIgralca	Število	Številka določenega igralca
StTurnir	Število	Številka turnirja
StDan	Število	Številka dneva za rezultat
TrenutniDatum	Datum	

TrenutniHDCP	Število	Hendikep pred turnirjem
NoviHDCP	Število	Hendikep po turnirju
StIgrisce	Število	Številka igrišča
Rezultat(18)	Število	Rezultati po luknjah
Fairway(18)	Število	Zadeti fairwayi po luknjah
GIR(18)	Število	Zadete zelenice po luknjah
Chip(18)	Število	Število chipov po luknjah
Bunker(18)	Število	Število udarcev iz bunkerja po luknjah
Putt(18)	Število	Število puttov po luknjah

6.2 Vnosne forme

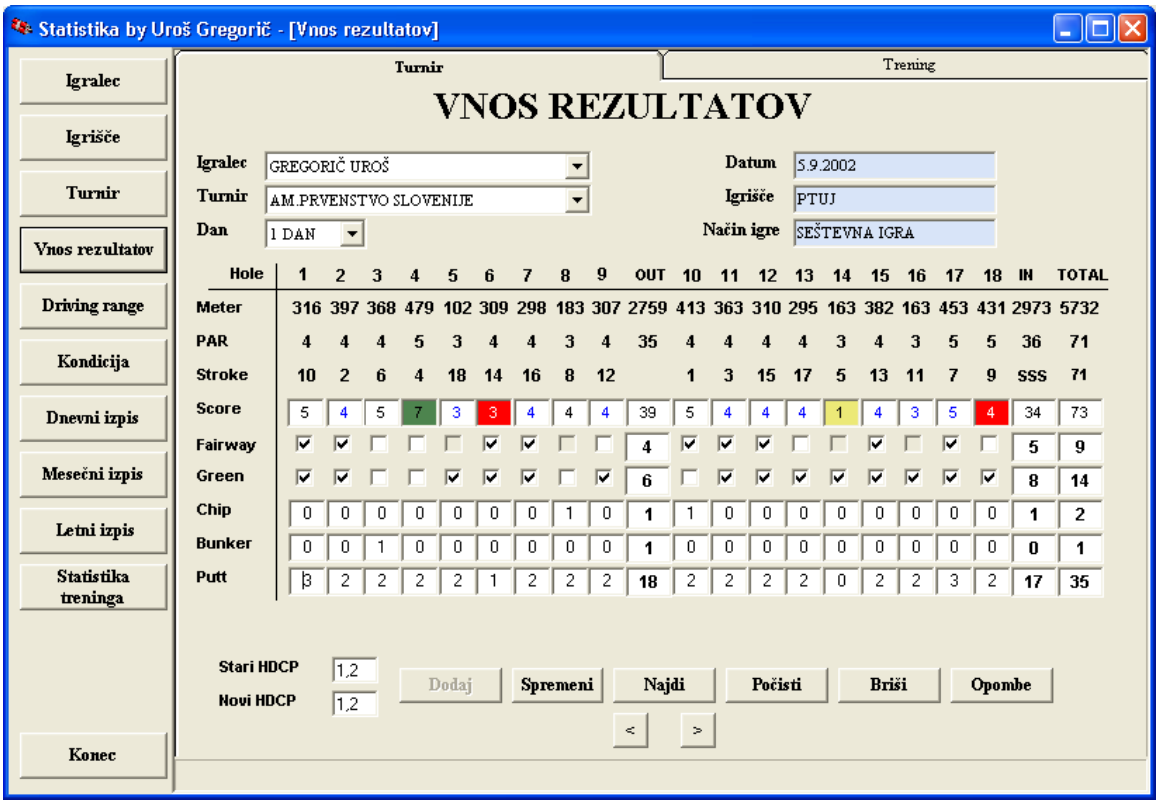
V zadnjih desetletjih se je tehnologija za vnos podatkov zelo spremenila. Kljub množici naprav je osnovno pravilo, da je kvaliteta izpisov iz informacijskega sistema odvisna od kvalitete vnosa. V fazi razvijanja vnosnih form, si sistemski analitik prizadeva razviti uporabniku prijazen vnos podatkov, hkrati zagotoviti kvaliteto in točnost podatkov. V tej fazi sistemski analitik izbere najboljšo strategijo, kako vnesti podatke v informacijski sistem (Shelly et al., 1998, poglavje 7, str. 1).

Vnosna oblika obsega metodo zajemanja podatkov, vnosa podatkov in obliko vnosnih form. V okviru oblikovanja vnosnih form sistemski analitik predvidi tudi način vnosa podatkov v bazo podatkov. Za dobro kvaliteto vnesenih podatkov lahko že med samim vnosom kontroliramo napake. Z določenimi kontrolami vnosa preprečimo napake, ki se pojavijo zaradi tipkarskih napak. Na te napake moramo uporabnika obvestiti ob vsakem napačnem vnosu in preden shranimo podatke v bazo podatkov.

Pri oblikovanju vnosnih form je pomembno, da je forma pregledna in enostavna za uporabo. Vse vnosne forme imajo naslednje lastnosti:

- Gumb »dodaj« zapiše nov zapis v bazo podatkov.
- Gumb »spremeni« spremeni podatke o obstoječem zapisu.
- Gumb »počisti« izprazni vsa polja na ekranu in pripravi forma za nov vnos.
- Gumb »briši« izbriše zapis iz baze podatkov.
- Gumb »najdi« poišče vse zapise, ki so že v bazi podatkov.
- Gumb »natisni« pripravi predogled tiskanja v orodju Word.
- Gumb »prikaži« izvede poizvedbo glede na postavljene kriterije in podatke prikaže na ekranu.
- Obstoječi zapisi so v seznamih. Z dvojnim klikom označimo zapis, ki bi ga želeli spreminjati ali brisati.
- Padajoči seznami nudijo izbiro med zapisi, ki so v bazi podatkov.

Slika 4: Forma za vnos rezultatov na turnirjih



Pri vnosu podatkov so predvidene določene kontrole, ki skrbijo, da so podatki čim bolj popolni in pravilni. V vseh formah se besedila vpisujejo avtomatično z velikimi črkami, kar naredi izpise veliko bolj pregledne in ne povzroča kasnejših problemov pri iskanju podatkov v bazi podatkov. V formi F_Igralec dobi uporabnik opozorilo v primeru nepopolnih podatkov. V formi F_Igrišče lahko uporabnik vnaša samo numerične podatke, v primeru alfanumeričnih se vrednosti takoj izbrišejo. Ker so dolžine za posamezno igrišče največkrat tri mestna števila, sem zaradi hitrejšega vnosa naredil, da kurzor preskoči v naslednje polje, ko so v prejšnjem vnesena tri števila. Pari igrišč so lahko samo eno mestna števila in težavnosti posameznih polj ne smejo presegati vrednosti 20. Pri vnosu podatkov o igrišču so osnovni podatki samo pari posameznih lukenj in SSS celotnega igrišča. S temi podatki je možno izračunati hendikep v primeru seštevne igre. Na tem mestu računalnik opozori uporabnika, da izračun hendikepa v primeru stableforda ni možen in mu da možnost ponovnega vnosa težavnosti. Naslednja forma zajema vnos podatkov. Tu je kontrola vnosa za rezultat, ki ne more presegati vrednosti 19, zato po vsakem vnosu števila, ki je različen od prve vrednosti 1 skoči v naslednje polje. Če igralec zadane t.i. Hole in One mora preskok potrditi z enter. V poljih za zadete zelenice in fairwaye so uporabniku namenjeni check-boxi, za chipe, bunker udarce in putte pa vnese s števili, ki ne morejo presegati vrednosti večje od 9. Uporabnika pri vnosu računalnik opozori,

če niso vneseni vsi rezultati oziroma mora v primeru treninga, določiti da je igral samo devet lukenj.

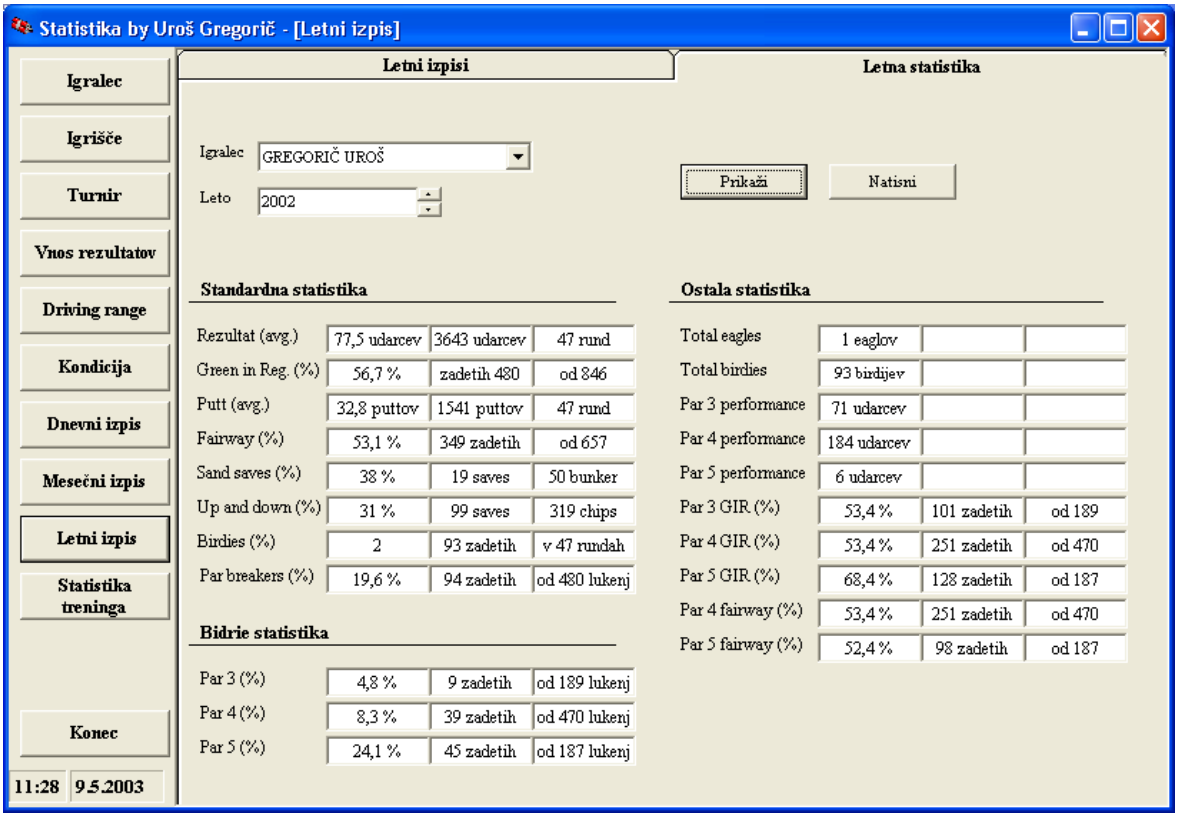
Slika 4 prikazuje formo za vnos rezultatov na turnirju. Zaradi boljše preglednosti se rezultati obarvajo glede na razliko od para igrišča in sicer rdeče za birdie, črno za bogy in zeleno za doble-bogy. Rezultati in posamezne statistike so sešteti po posameznih devetih luknjah. Ta oblika prikaza je standardna za prikaz rezultatov s turnirja.

6.3. Izpisi

Uporabniki vidijo uporabnost informacijskega sistema v njegovi zmožnosti dajanja pravih informacij za opravljanje njihovega dela. Ne zanimajo jih tehnične specifikacije. Večina bo na podlagi končnih izpisov ocenjevala kvaliteto celotnega informacijskega sistema. Eden od ključnih dejavnikov uspeha novega informacijskega sistema je v tem, ali informacijski sistem resnično podpira potrebe uporabnikov in organizacije. Če želimo ustvariti dober sistem, morajo izpisi biti uporabni, natančni, razumljivi in pravočasni (Shelly et al., 1998, poglavje 6, str. 3).

Nekaj zadnjih form je namenjenih izpisom in statistiki turnirjev in treninga. Najprej so predstavljeni dnevni izpisi. V okviru dnevnih izpisov imajo uporabniki na voljo izpise za posamezne turnirske dneve, za posamezne turnirje in turnirsko statistiko. V mesečnih izpisih si lahko uporabnik ogleda rezultate za določen mesec in njihovo statistiko. Zadnji turnirski izpis je namenjen letnim pregledom rezultatom. Na tem mestu ima uporabnik več možnosti. Lahko izbira med vsemi rezultati, ki so v bazi, ali samo za določenega igralca, za določen golf klub ali za določen turnir. Hkrati lahko izbirajo tudi 15 najboljših rezultatov in 5 najslabših rezultatov v določenem obdobju. V okviru letne statistike imajo uporabniki tudi zelo popolno analizo turnirjev, ki jo lahko primerjamo z analizo najboljših igralcev golfa na svetu. Vse posamezne izpise je možno tudi natisniti v papirno obliko. Zadnji izpisi se navezujejo na pregled analize treninga. Tudi tu ima uporabnik razdeljene statistike za različna obdobja. Na tem mestu je nekoliko večji poudarek na podatkih o samem treningu in nekaj manj na statistiki iz igrišča.

Slika 5: Prikaz izpisa letnih podatkov za določenega igralca

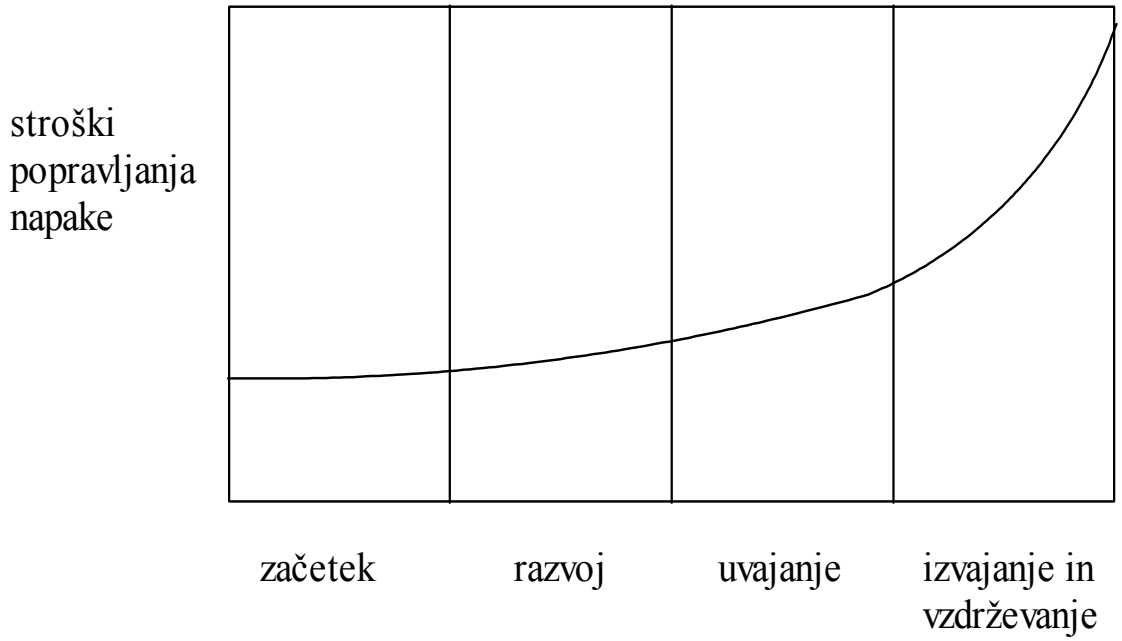


7. Razvoj informacijskega sistema

V tretji fazi življenjskega cikla je sistemski analitik oblikoval obliko informacijskega sistema. V četrti fazi, to je razvoj informacijskega sistema, služi specifikacija oblike kot osnovno gradivo za razvoj informacijskega sistema. Na tem mestu programerji planirajo, razvijajo, dokumentirajo, integrirajo in testirajo nov informacijski sistem in njegove module. Kadar je informacijski sistem kupljen, mu programerji dodajo le posebne funkcije (Shelly et al., 1998, poglavje 10, str. 3-4).

V današnjem konkurenčnem poslovnem okolju, si podjetja prizadevajo zagotoviti čim večjo kvaliteto svojim izdelkom in storitvam. Ne glede na to, kako skrbno je sistem oblikovan, se pojavljajo napake. Podrobna testiranja bodo pokazala napake v končnih fazah razvoja, ampak še vedno ceneje kot odkrivanje napak s strani končnih uporabnikov. Slika 6 prikazuje stroške odkrivanja napak. Vsako odpravljanje težav v prvih fazah razvoja informacijskega sistema je neprimerno ceneje, kot kasnejša popravljanja.

Slika 6: Stroški za popravljanje začetnih napak glede na čas odkritja



Vir: Gradišar et al,. 1996, str. 424.

7.1. Razvoj informacijskega sistema

Razvoj informacijskega sistema obsega planiranje, programiranje in testiranje. Pri planiranju sistema, večina analitikov uporablja sistem od zgoraj navzdol, ki predvideva najprej celotni pristop, kasneje pa razkrije še posamezne detajle. Sistemski analitik definira glavne lastnosti informacijskega sistema in jih nato razbija na posamezne module. Modul je določena programska koda, ki jo je možno izvajati kot skupina. Tako lahko posamezni programerji razvijajo različne module istočasno (Shelly et al., 1998, poglavje 10, str. 6-7).

Programiranje je proces, v katerem programer spreminja programsko logiko v specifična navodila, ki so lahko izvršljiva v računalniku. Kadar je analitik dobro pripravil obliko informacijskega sistema, je programiranje enostaven proces pretvorbe logičnih funkcij v programsko kodo. Za pisanjem kode, pride na vrsto testiranje informacijskega sistema. Najprej odpravimo sintaktične napake, te se pojavijo zaradi tipkarskih napak. Naslednjo vrsto napak je nekoliko težje odkriti, to so logične napake. Logične napake povzročijo, da nam sistem vrne nepravilne rezultate. Ko programer testira podatke, najprej testira posamezne enote. Ko odpravi napake po posameznih enotah, lahko testira napake med posameznimi povezavami med enotami in na koncu je potrebno testirati delovanje celotnega sistema. Testiranja vedno

izvajamo na provizoričnih podatkih, ki imajo podobne vrednosti, kot jih bodo imeli kasnejši pravi podatki (Shelly et al., 1998, poglavje 10, str. 9-11).

Za pisanje programske kode sem uporabil orodje Visual Basic. Visual Basic 6.0 (VB) je eno od orodij iz skupine Microsoft Visaul Studio 6.0. VB je razvojno okolje, ki združuje več funkcij razvijanja programa, na primer urejanje grafičnega vmesnika, urejanje kode, prevajanje, poskusno izvajanje (»razhroščevanje«) itn. Primeren je tudi za povezovanje z drugimi programskimi rešitvami in za povezovanje z bazami podatkov.

Celoten informacijski sistem sem razdelil na posamezne enote, ki obsegajo posamezne obrazce. Nato sem definiral izvajanje posamezne enote v okviru celotnega sistema in posameznim enotam dodelil še bolj detajlne lastnosti. Tako sem dobil posamezne parcialne enote za katere sem začel razvijati programsko kodo. Pri pisanju kode sem si pomagal z diagramom entitet-povezav, fizičnim modelom, vnosnimi formami in želenimi izpisi.

Pri pisanju kode sem se držal določenih pravil, ki omogočajo lažjo preglednost kode in hitrejše odkrivanje napak. Že v bazi podatkov so imenovane tabele s predpono T, in poizvedbe s predpono Q, kar jasno pokaže, s čim imamo opravka. Problem se pojavi predvsem takrat, ko imamo tabele in poizvedbe poimenovane z enakimi imeni. Ravno tako je imenovanje pomembno v Visual Basicu. Zaradi velikega števila objektov je potrebno vsak objekt poimenovati, tako kasneje programer ve, na kateri objekt se nanaša koda. Mogoče se zdi na začetku zamudno opravilo, vendar brez te stopnje kasnejše programiranje ni možno. Za lažjo preglednost posameznim objektom dodelim še vrsto objekta, tako polje z besedilom označim z txt, combo-box s cmb in gumb z cmd. Pri pisanju kode lahko programer z zamikanjem vrstic napravi boljši pregled nad kodo. S komentarji, ki so napisani ob kodi, lahko skrajšamo iskanje ob morebitnih težavah. To velja predvsem, če je za vzdrževanje programa odgovorna druga oseba. Ker se določeni odseki kode ponavljajo večkrat v celotni programski rešitvi, te dele zapišemo kot samostojne procedure. Vsakič ko to proceduro potrebujemo, jo samo pokličemo. Na ta način rešimo dve težavi. Najprej je potrebno manjše število programske kode in tako pospešimo razvoj programske rešitve. Druga še bolj pomembno dejstvo, pa je popravljanje kasnejše kode. Če imamo kodo zapisano samo na enem mestu, jo popravimo na enem mestu in spremembe veljajo na vseh mestih hkrati.

Funkcija za izračun hendikepa je zapisana v določenem modulu. Izračun je prilagojen določenemu načinu igre. Programska koda za izračun hendikepa v primeru seštevne igre je v prilogi številka 3. Funkcija tako izračuna novi hendikep in nam vrednost vrne nazaj na našo formo. Lastnost funkcij je, da nam vrne eno samo vrednost.

7.2. Uvedba

Programerji razvijajo nove informacijske sistema v testnem okolju. To okolje je ločeno od operativnega okolja, kjer delujejo s pravimi podatki. Ravno ločenost okolij zagotovi integriteto in varnost podatkov. Dostop do operacijskega okolja je omejen na uporabnike. Sistemski analitik in programer naj ne bi imela dostopa do pravih podatkov, razen v izjemnih primerih. Testno okolje vsebuje kopije vseh programov, procese in testne podatke. Preden se odločimo za spremembe v operativnem okolju, moramo spremembe testirati v testnem okolju in pridobiti dovoljenje uporabnikov (Shelly et al., 1998, poglavje 10, str. 10-12).

Kadar želimo pripraviti operativno okolje, moramo skrbno pregledati programsko in strojno opremo, operacijski sistem, mrežne povezave in vse ostale informacijske sisteme, ki lahko vplivajo na delovanje sistema.

Informacijski sistem za obdelavo rezultatov deluje na vseh programskih okoljih, ki deluje na Windows operacijskih okoljih. Trenutno povezava z internetom ni pogoj za delovanje programske rešitve. Informacijski sitem deluje kot samostojen program. Baza podatkov je narejena v Accessu in deluje tudi v primeru, če računalnik nima nameščenega orodja Access. Informacijski sistem bodo uporabniki namestili na svoj računalnik s pomočjo namestitvenega programa, kjer jih bo čarovnik vodil skozi namestitev. Za kasnejšo uporabo bodo uporabljali uporabniki bližnjico do programske rešitve, ki bo shranjena na njihovem namizju.

7.2.1. Izobraževanje in način prehoda

Noben sistem ne more biti uspešno uveden, če ni primernega izobraževanja. Za uspešno uporabo informacijskega sistema je potreben trening za vse uporabnike in tudi nadrejene. Uporabniki lahko zavrnejo uporabo sistema, če ne razumejo delovanja sistema in poznavanja njegovih prednosti (Shelly et al., 1998, poglavje 11, str. 3).

Glavne tri skupine, ki potrebujejo izobraževanje, so uporabniki, managerji in oddelek za informatiko. Managerji ne potrebujejo znanj o tehnični specifikaciji delovanja sistema. Na drugi strani moramo uporabnike izobraziti tako, da lahko sistem uporabljajo pri vsakodnevnih opravilih. Zaposleni v informacijskem oddelku verjetno potrebujejo največ podatkov. Da bodo nudili uspešno podporo novemu sistemu, morajo poznati sistemske funkcije, kako sistem podpira poslovanje podjetja in opravila, ki jih opravljajo posamezni uporabniki. Za bolj učinkovito predstavitev novega informacijskega sistema, lahko novosti predstavimo v obliki seminarjev, ki pa potekajo ločeno za posamezne skupine (Shelly et al., 1998, poglavje 11, str. 3-4).

V konkretnem primeru bodo uporabniki novega informacijskega sistema razpršeni po vsej Sloveniji. Vsako usklajevanje terminov in predstavitev bi bila skoraj nemogoča. Ker v reviji Golf stalno sodelujem z rubriko Golf je šport, sem se odločil, da predstavitev novega sistema opišem v tej rubriki. Revijo dobijo skoraj vsi golf igralci na dom in tako je zavzeta kar najširša publika. V članku bom predstavil bistvene značilnosti programske rešitve in izdelal kratka navodila za uporabo. Način uporabe je dokaj poenoten skozi celoten informacijski sistem in zaradi dobrega vizualnega izgleda predvidevam, da večjih težav z uporabo novega informacijskega sistema ne bi smelo biti. V raziskavi, ki sem jo izvedel, sem ugotovil, da imajo bodoči uporabniki že dobra znanja o uporabi računalnika, to pa je tudi vse, kar se od njih pričakuje.

V fazi prehoda se opravi dva procesa. Prvi proces je prehod na nov sistem, drugi proces je prilagoditev podatkov iz starega sistema v nov sistem. Čeprav je včasih možno prenos podatkov avtomatizirati, nov sistem potrebuje še dodatne podatke. Vnos teh podatkov je lahko zamudno opravilo. Pri vsakem prenosu podatkov so potrebne stroge kontrole vnosa podatkov, saj so tu podatki izredno ranljivi. Predvsem moramo sistem zaščititi pred nepooblaščenimi vdori v sistem (Shelly et al., 1998, poglavje 11, str. 10).

V literaturi lahko srečamo različne prehode na nov informacijski sistem. Prehod je lahko hiter ali počasen odvisno od metode prehoda. Pri prehodu na nov sistem je časovna usklajenost izredno pomembna. Večina uporabnikov si ne želi sprememb in bodo zato slušali izkoristiti vsako napako pri prehodu sistema, za njegovo neuporabo v praksi (Gradišar et al., 2001, str. 427).

Neposredni prehod predvideva, da nov informacijski sistem začnemo uporabljati in hkrati opustimo starega. Ta način prehoda je najcenejša oblika prehoda, vendar pa nosi tudi največje tveganje za neuspeh. Uporabimo ga takrat, ko postopen prestop ni možen ali kadar starega sistema ni bilo. Vzporedno delovanje vključuje delovanje starega in novega sistema istočasno. Pri takšnem uvajanju imajo uporabniki dvojno delo, vendar pa je najbolj varen. V primeru napak, lahko nov informacijski sistem nadomestimo s starim. Pilotno uvajanje predvideva, da z novim informacijskim sistemom začnejo najprej samo določeni uporabniki. Ko v podjetju vidijo, da je informacijski sistem zanesljiv, se odločijo za popolno uvedbo novega sistema. Postopni prehod vključuje nov sistem po stopnjah. Namesto da vključite celotni sitem naenkrat, se uvede samo določen del in kasneje še ostale dele. V postopnem prehodu dodelimo nov informacijski sistem vsem uporabnikom, in ne tako kot pri pilotnem prehodu, kjer z novim sistemom začnejo samo določeni uporabniki. Z vidika stroškov je najdražji vzporedni prehod, nato sledita postopni in pilotni prehod in najcenejši je neposredni prehod (Gradišar et al., 2001, str. 427).

Zaradi dejstva, da je informacijski sistem nov za večino uporabnikov, sem predvidel neposredni prehod. Da bi zmanjšal tveganje nedelovanja informacijskega sistema, sem informacijski sistem že sam nekaj časa pilotno uporabljal in odpravljal njegove pomanjkljivosti. Kljub testiranju pričakujem, da bo do določenih težav pri prvih uporabnikih prišlo. Ker sem z uporabniki v dobrem odnosu, pričakujem, da bomo v skupno dobro vse napake čim preje odpravili. Pri prehodu večina uporabnikov ne bo vnašala podatkov za prejšnja leta. Analize in statistike so predvsem namenjene ugotavljanju sedanjega stanja in možnih izboljšav treninga v prihodnosti. S pomočjo ankete sem ugotovil, da trenutno uporabljata samo dva programsko rešitev za analizo treninga. Zaradi težavnosti prenosa podatkov, prenos starih podatkov ne bo mogoč. Po drugi strani pa je trenutna rešitev namenjena vsem igralcem in zato zahteva bistveno več podatkov. Uporabniki bi morali ravno tako še dodajati podatke za posamezne turnirje in treninge.

8. Vzdrževanje informacijskega sistema

Zadnja faza v metodi življenjskega cikla je vzdrževanje informacijskega sistema. Ta faza se začne, ko sistem postane operativen in se zaključi, ko sistem nehamo uporabljati. Čeprav se na prvi pogled zdi ta faza najlažja, pa v praksi kar 70 % informatikov vzdržuje informacijske sisteme (Gradišar, 2003).

V fazi vzdrževanja informacijskega sistema so stroški razporejeni v obliki črke u. Celotne stroške sistema lahko ločimo na dve skupini. Prva so operativni stroški, kot stroški najema strojne in programske opreme. Druga skupina so stroški vzdrževanja. Operativni stroški se skozi obdobje ne spreminjajo in ostajajo vseskozi na enakem nivoju. Drugače je s stroški vzdrževanja. V začetku uporabljanja sistema so stroški vzdrževanja izredno visoki. Z izboljšavami in popravki, programer izboljša sistem in tako se stroški znižujejo. Tako ostanejo kar nekaj časa. Ko se začne okolje spreminjati, se začnejo spreminjati tudi potrebe informacijskega sistema. Programerji morajo svoje programe prilagajati spremembam v okolju. V tej fazi stroški vzdrževanja naraščajo. Ko so stroški vzdrževanja večji, kot stroški razvoja novega sistema, se obstoječi sistem opusti in zamenja z novim (Shelly et al., 1998, poglavje 12, str. 7).

V literaturi lahko zasledimo, da je vzdrževanje sistema razdeljeno na tri vrste. Prva vrsta je popravno vzdrževanje (angl corrective maintenance), kjer odkrijemo in popravimo napake. Prilagodljivo vzdrževanje (angl. adaptive maintenance) doda izboljšave obstoječemu sistemu. Z izboljšavami dosežemo večjo učinkovitost in lažje vzdrževanje sistema. Izpopolnjevalno vzdrževanje (angl. perfective maintenance) obsega spremembe v informacijskem sistemu, da postane bolj zmogljiv ali zanesljiv (Shelly et al., 1998, poglavje 12, str. 6-7).

Vzdrževanje informacijskega sistema je zelo pomembna faza v metodi življenjskega cikla. Predvsem v začetni fazi uporabe sistema bo prihajalo, do največjih težav. V ta namen sem v programsko rešitev, vgradil posebno funkcijo, ki mi preko elektronske pošte pošilja napake. Poleg opisa napake, dobim tudi številko napake in pri katerem dogodku se je napaka zgodila ter v kateri formi. Da lahko dobivam pošto mora vsak uporabnik vnesti podatke o odhajajoči pošti (angl. outgoing mail) in številki vrat (angl. port). Ko programska rešitev povzroči več kot dve napaki, program avtomatsko pošlje napake na moj elektronski naslov. S sodelovanjem z uporabniki bom lahko sistem popravil tako, da bo primeren za uporabo.

9. Zaključek

Diplomska naloga pojasnjuje razvoj programske rešitve z metodo življenjskega cikla. S pomočjo petih faz, ki jih vsebuje metoda, je bila sočasno razvita tudi programska rešitev, ki podpira potrebe analize treninga in turnirjev golf igralcev. Program lahko uporabljajo vsi igralci, saj si vsak posameznik lahko prilagodi izpise glede na njegove potrebe. Program omogoča izračun rezultatov po dveh vrstah igre, to sta seštevna igra in stableford, ter dnevne, mesečne in letne statistike turnirjev in treningov.

Čeprav je metodologija življenjskega cikla primerna predvsem za velike informacijske sisteme, se je v konkretnem primeru metoda izkazala kot primerna. V vsaki fazi je potrebno opraviti določene zahteve, da lahko nadaljuješ z naslednjo fazo. Metoda življenjskega cikla v začetnih fazah zahteva od analitika, da dobro prouči potrebe uporabnikov in da te potrebe prezrcali v kasnejšo programsko rešitev. Ravno te prve faze so tudi izredno zamudne. Dobro opravljeno delo se obrestuje pri kasnejšem pisanju programske kode. Kljub podrobnemu opravljanju posameznih faz, se je včasih potrebno vrniti fazo ali dve nazaj. Napake in nejasnosti je potrebno čim hitreje odpraviti tudi z vidika stroškov razvoja. Na primer, ko je bilo potrebno napisati programsko kodo, je bilo potrebno napraviti korekture v podatkovnem modelu. Za vodenje zgodovine spreminjanja hendikepa, je bilo potrebno v bazo zapisati tako začetni kot tudi spremenjeni hendikep. Zadnja faza, to je vzdrževanje sistema, bo nastopila, ko bodo sistem začeli uporabljati.

Z metodo življenjskega cikla je razvoj programske rešitev sistematičen in zahteva od razvijalca veliko časa v prvih korakih razvoja. Mogoče je bilo tu meni delo nekoliko olajšano, saj izredno dobro poznam naravo problema in sam proces dela. Vseeno sem želel dobiti čim več zahtev od bodočih uporabnikov in z anketo in intervjuji pridobil želje uporabnikov. Ko imamo izdelan logični in fizični model, lahko začnemo razvijati informacijski sistem. Četrti korak, ki obsega tudi pisanje programske kode, je najobsežnejši del razvoja informacijskega sistema. Predvsem je potrebno veliko časa nameniti kontroli pri vnosu in zaščiti programske rešitve. Uporabnik je z opozorili obveščen o naslednjem koraku pri uporabi programske rešitve.

Za razvoj programske rešitve sem potreboval šest mesecev in v mesecu juniju 2003 je bila programska rešitev dokončana. V začetku julija je bila programska rešitev na voljo vsem igralcem golfa. Ravno tako je bil v mesecu juliju objavljen članek v rubriki Golf je šport, v katerem je bila predstavljena programsko rešitev. Na dveh straneh so bila predstavljena navodila za namestitev programa na osebni računalnik in navodila za uporabo. Tako lahko vsak uporabnik pogleda v navodila in uporablja programsko rešitev.

Po začetni oceni nova programska rešitev dobro pokriva potrebe igralcev golfa in je primerna za uporabo. Program bo na voljo vsem igralcem brezplačno. Vsi zainteresirani ga bodo dobili na internet strani in si ga bodo namestili na osebni računalnik. Z izpisi bodo lahko bolje analizirali svojo igro in planirali svoj trening. Izpisi in statistike bodo pomagali tudi profesionalnim učiteljem pri oblikovanju treningov.

10. Literatura

[1] Brcar Franc: Prenova in informatizacija poslovanja z metodo poenoteni jezik modeliranja. Magistrsko delo. Ljubljana : Ekonomska fakulteta, 2002. 84 str.

[2] Campbell Malcolm: Najnovejše tehnike v golfu. Ljubljana : Slovenska knjiga d.o.o., 1997. 216 str.

[3] Grad Janez, Kovačič Andrej: Osnove baze podatkov in njene uporabe. Ljubljana : Ekonomska fakulteta, 1996. 120 str.

[4] Gradišar Miro, Resinovič Gortan: Informatika v poslovnem okolju. Ljubljana : Ekonomska fakulteta, 2001. 508 str.

[5] Jaklič Jurij: Upravljanje in uporaba podatkov. Ljubljana : Ekonomska fakulteta, 2002. 213 str.

[6] Kovačič Andrej: Informatizacija poslovanja. Ljubljana : Ekonomska fakulteta, 1998. 214 str.

[7] Kovačič Andrej, Vintar Mirko: Načrtovanje in gradnja informacijskih sistemov. Ljubljana : DZS, 1994. 316 str.

[8] Kroenke M. D.: Database Processing: Fundamentals, Design & Implementation. USA : Prentice-Hall, 2000. 601 str.

[9] Shelly Gary, Cashman Thomas, Rosenblatt Harry: Systems Analysis and Design. USA : International Thomson Publishing, 1998. 687 str.

[10] Turban Efraim, Lee Jae, King David, Chung Michael: Electronic Commerce A Managerial Perspective. New Jersey : Prentice Hall Inc, 2000. 518 str.

[11] Turban Efraim, McLean Ephraim, Wetherbe James: Information Technology for Management. USA : John Wiley & Sons, 1999. 791 str.

[12] Watson T. Richard: Data Management: Databases and Organizations. USA : John Wiley & Sons, 1999. 604 str.

11.Viri

[1] Grad Anton, Škerlj Ružena, Vitorovič Nada: Angleško-slovenski slovar. Ljubljana : DZS, 1984. 657 str.

[2] Terminološki slovar informatike. [http://www.ef.uni-lj.si/terminoloskislovar/index.asp], 15. marec 2003.

[3] USPGA Players Stats. [URL: http://www.pgatour.com/players/00/87/93/stats.html], 28. marec 2003.

[4] Domača spletna stran GZS. [URL: http://www.golfportal.info], 15. april 2003.

[5] Gradišar Miro: Razvoj informacijskih sistemov, zapiski predavanj 2002/2003

Slovarček informacijskih izrazov

Adaptive maintenance – prilagodljivo vzdrževanje
Corrective maintenance – popravno vzdrževanje
Database management system – DBMS – sistem za upravljanje baz podatkov
Data definition language – DDL – jezik za definiranje podatkov
Data manipulation language – DML – jezik za manipulacijo podatkov
Data dictionary – podatkovni slovar
Desicion support systems – sistemi za podporo odločanju
Economic feasibility – ekonomska izvedljivost
Executive information system – informacijski sistemi za direktorje
Fact-finding – zbiranje zahtev
Feasibility study – študija izvedljivosti
Garbage-in, garbage-out – GIGO – smeti noter, smeti ven
Groupware systems – sistemi za podporo dela v skupini
Local area network – LAN – lokalno omrežje
Management information systems – informacijski sistem za upravljanje
Office automation system – pisarniški sistemi
Operational feasibility – operativna izvedljivost
Operational systems – izvajalni operacijski sistem
Outgoing mail – odhajajoča pošta
Perfective maintenance – izpopolnjevalno vzdrževanje
Port - vrata
Preliminary investigation report – preliminarno poizvedbeno poročilo
Query language – poizvedbeni jezik
Structured query language – strukturiran poizvedbeni jezik
System design specification – specifikacija oblike informacijskega sistema
Systems development life cycle – metoda življenjskega cikla
System implementation – razvoj informacijskega sistema
Systems request – sistemska zahteva
System requirements document – sistemske zahteve
Technical feasibility – tehnična izvedljivost
Tunnels – predor
Utilities programs – uporabniški programi
Watterfall model – model slap

Slovarček uporabljenih izrazov golfa

Bližanje (angl. putt) udarec, ki ga igralec igra na zelenici.

Bogi (angl. bogey) izraz za seštevek udarcev, ki je za en udarec višji, kot je par luknje.

Bruto rezultat (angl. gross score) je seštevek vseh udarcev s katerimi je igralec zaključil igro oziroma dogovorjeno rundo.

Čistina (angl. fairway), nizko pokošena površina med odbijališčem in zelenico. Čistina navadno meji na malo višje košeno travo.

Čip (angl. chip) udarec, ki ga igralec igra blizu zelenice proti luknji.

Gor in noter (angl. up and down) izraz, ki se uporablja takrat, ko igralec izvede en kratek bliževalni udarec s trave in z enim puttom zadene luknjo.

Hendikep (angl. handicap), način ocenjevanja igralčevih sposobnosti v razmerju do para igrišča. Tak način izračunavanja omogoča igralcem različnih sposobnosti, da igrajo drug proti drugemu na teoretično enakih izhodiščih. Hendikep navadno temelji na povprečju seštevkov igralčevih udarcev v primerjavi s standardom igrišča.

Neto rezultat (angl. nett score) je seštevek igralčevih udarcev v eni rundi po odštetju hendikepa. Neto rezultat je osnova za računanje hendikepa.

Odbijališče (angl. tee) je ravno, nizko pokošeno območje, največkrat malo dvignjeno, s katerega se izvaja prvi udarec na vsaki luknji. Prostor za prvi udarec je pravokotnik, katerega sprednjo stranico določata dva označevalca, širino pa se določi z dolžino dveh palic.

Orel (angl. eagle) število udarcev, ko igralec konča luknjo z dvema udarcema manj, kot je njen par.

Par (angl. par) ocenjeni standardni seštevek udarcev posamezne luknje, ki temelji na dolžini luknje in številu udarcev, kolikor jih potrebuje prvovrsten igralec, da v normalnih razmerah konča igro na luknji in igrišču.

Ptička (angl. birdie) izraz za število udarcev, ki je en udarec nižji, kot je par luknje.

Prvih devet (angl. out) prva skupina devetih lukenj na golf igrišču z 18 luknjami.

Seštevek udarcev (angl. score) število udarcev , ki jih je igralec dosegel na eni luknji.

Seštevna igra (angl. stroke play) je oblika igre, pri kateri se seštejejo vsi udarci na posameznih luknjah in morebitni kazenski udarci. Seštevek udarcev vseh lukenj runde velja kot seštevek dogovorjene runde.

Stableford oblika tekme, pri kateri se seštevajo točke, ki jih igralec dobi glede na seštevek udarcev na posamezni luknji.

Standard Scratch Score (SSS) ocena za par za posamezno igrišče, ki je osnova za izračunavanje hendikepa.

Težavnostna lista lukenj (angl. Stroke), lista, ki označuje vrstni red lukenj po težavnosti igranja.

Zadnjih devet (angl. in) druga skupina devetih lukenj na golf igrišču z 18 luknjami.

Zelenica (angl. green) je območje na igrišču, posebej pripravljeno za puttanje, z nizko pokošeno travnato površino, v katero je vrezana luknja.

PRILOGA

1. Anketa

Spol: M Ž HDCP:_____
Starost:_____ Datum:_____

Moje ime je Uroš Gregorič in sem v četrtem letniku Ekonomske fakultete. Izdelal bom diplomsko nalogo, ki bo analizirala stanje uporabe programskih rešitev pri treningu in turnirjih golf igralcev v Sloveniji. Pridobljeni rezultati ankete bodo osnova za izdelavo programske rešitve. Podatki bodo uporabljeni samo za izdelavo diplomske naloge.

Vsak odgovor skrbno preberite in obkrožite črko pred odgovorom, s katerim se strinjate. Pri posameznih vprašanjih je možnih več odgovorov! Prosim tudi tiste, ki boste odgovorili na vprašanje št. 3 z ne, da mi pošljete anketo nazaj? Hvala za vaš trud in sodelovanje.

Anketa

1. Ali imate osebni računalnik?
   a) da
   b) ne

2. Ali že uporabljate program za statistiko in analizo vašega treninga in turnirjev? Če odgovorite z da, napišite katerega!
   a) da, _____
   b) ne

3. Ali ste pripravljeni uporabljati računalniški program, ki bi analiziral vaš trening in turnirje? Če ste odgovorili z ne, vam **ni potrebno** odgovarjati na naslednja vprašanja!
   a) da
   b) ne

4. Kateri operacijski sistem uporabljate na vašem osebnem računalniku?
   a) Windows
   b) Linux
   c) Ostalo, _____

5. Ali imate povezavo z internetom?
   a) da

b) ne

6. Koliko časa ste pripravljeni nameniti vnosu podatkov?
   a) 0 minut
   b) 1-2 minute
   c) 3-5 minut
   d) 6-10 minut

7. Kateri podatki vas zanimajo o vaši igri? Možnih je več odgovorov!
   a) HDCP
   b) število udarcev po luknjah
   c) število puttov po luknjah
   d) število zadetih fairwayev
   e) število zadetih greenov
   f) število UP and DOWN na krog (v primeru, da imamo en chip in en putt)
   g) število SAND SAVES na krog (iz bunkerja naredimo en udarec in en putt)
   h) letno povprečno število udarcev
   i) letno povprečno število puttov na krog
   j) letni povprečni odstotek zadetih fairwayev
   k) letni povprečni odstotek zadetih greenov
   l) ostalo (napišite vaše želje)_____

8. Ali bi želeli imeti podatke tudi o vašem treningu na vadišču? Če odgovorite z ne, vam ni potrebno odgovoriti na vprašanje št. 9 in nadaljujete pri vprašanju 10.
   a) da
   b) ne

9. Katere kategorije bi vas zanimale? Izberite med varianto I. ali II. in obkrožite I ali II.!
   I. a) puttanje
      b) kratka igra - trava
      c) kratka igra -bunker
      d) kratka železa
      e) srednja železa
      f) dolga železa
      g) lesovi
      h) ostalo, _____
   II. a) puttanje
      b) kratka igra
      c) železa
      d) lesovi

e) ostalo, _____

10. Kako pogosto bi želeli imeti izpise rezultatov? Možnih je več odgovorov!
    a) dnevno
    b) mesečno
    c) letno
    d) ostalo,_____

11. Kako bi želeli imeti predstavljene podatke? Možnih je več odgovorov!
    a) v tabeli
    b) v grafu
    c) ostalo (napišite svoje želje)_____

12. Ali bi uporabljali program, če bi morali vnašati podatke preko Interneta?
    a) da
    b) ne

13. Ali imate dlančnik?
    a) da
    b) ne

14. Ali bi uporabljali dlančnik za vnos podatkov?
    a) da
    b) ne

15. Ali bi uporabljali dlančnik že med samo igro?
    a) da
    b) ne

16. Kateri način vnosa bi vam bil najbolj priljubljen? Možnih je več odgovorov!
    a) preko osebnega računalnika, dostop je možen samo na vašem računalniku
    b) z internetom preko osebnega računalnika, vnos je možen na vseh računalnikih, ki imajo internet
    c) preko dlančnika

Hvala za sodelovanje

2. Opis ostalih tabel v fizičnem modelu

Tabela 1: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Odbijalisca

Ime polja	Podatkovni tip	Opis
StOdbijalisce	Samoštevilo	Zaporedno število odbijalisca
Odbijalisce	Besedilo	Opis začetnih odbijališč

Tabela 2: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_ImeIgrisca

Ime polja	Podatkovni tip	Opis
StImeIgrisca	Samoštevilo	Zaporedno število ime igrišča
ImeIgrisca	Besedilo	Ime igrišča

Tabela 3: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_igrisce

Ime polja	Podatkovni tip	Opis
StIgrisce	Samoštevilo	Zaporedno število igrišča
StImeIgrisca	Število	Številka določenega imena igrišča
StOdbijalisce	Število	Številka določenega odbijališča
SSS	Število	Standard igrišča
Par(18)	Število	Par na posamezni luknji
Meter(18)	Število	Dolžina na posamezni luknji
Stroke(18)	Število	Težavnost na posamezni luknji

Tabela 4: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Trening

Ime polja	Podatkovni tip	Opis
StTrening	Samoštevilo	Zaporedno število treninga
StIgralec	Število	Številka določenega igralca
Datum	Datum	
StIgrisce	Število	Številka igrišča
Rezultat(18)	Število	Rezultati po luknjah
Fairway(18)	Število	Zadeti fairwayi po luknjah
GIR(18)	Število	Zadete zelenice po luklnjah
Chip(18)	Število	Število chipov po luknjah
Bunker(18)	Število	Število udarcev iz bunkerja po luknjah
Putt(18)	Število	Število puttov po luknjah

Tabela 5: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Driving_Range

Ime polja	Podatkovni tip	Opis
StDrivingRange	Samoštevilo	Zaporedno število treninga na vadišču
StIgralec	Število	Številka določenega igralca
Datum	Datum	
Putt	Število	Čas treninga
KratkaIgraTrava	Število	Čas treninga
KratkaIgraBunker	Število	Čas treninga
KratkaZeleza	Število	Čas treninga
SrednjaZeleza	Število	Čas treninga
DolgaZeleza	Število	Čas treninga
Driver	Število	Čas treninga
Lesovi	Število	Čas treninga

Tabela 6: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Kondicija

Ime polja	Podatkovni tip	Opis
StKondicija	Samoštevilo	Zaporedno število vadbe kondicije
StIgralec	Število	Številka določenega igralca
Datum	Datum	
Tek	Število	Čas treninga
Kolo	Število	Čas treninga
Plavanje	Število	Čas treninga
Utezi	Število	Čas treninga
HrbetTrebuh	Število	Čas treninga
Streching	Število	Čas treninga

Tabela 7: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Klub

Ime polja	Podatkovni tip	Opis
StKlub	Samoštevilo	Zaporedno število kluba
Klub	Besedilo	Ime kluba

Tabela 8: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_NacinIgre

Ime polja	Podatkovni tip	Opis
StNacinIgre	Samoštevilo	Zaporedno število načina igre
NacinIgre	Besedilo	Naziv načina igre

Tabela 9: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Dan

Ime polja	Podatkovni tip	Opis
StDan	Samoštevilo	Zaporedno število dneva
Dan	Besedilo	Dan

Tabela 10: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Tees

Ime polja	Podatkovni tip	Opis
StTees	Samoštevilo	Zaporedno število odbijališča
Tees	Besedilo	Opis odbijališča

Tabela 11: Prikaz atributov, podatkovnih tipov in opisov v tabeli T_Spol

Ime polja	Podatkovni tip	Opis
StSpol	Samoštevilo	Zaporedno število spola
Spol	Besedilo	Spol

3. Funkcija za izračun hendikepa v primeru seštevne igre

```
Public Function IzracunHdcpSestevna(Hdcp As Single)

 Dim intHdcp As Integer
 Dim intPrvihDevet As Integer
 Dim intDrugihDevet As Integer
 Dim intBrutto As Integer
 Dim intNetto As Integer
 Dim N As Long
 Dim intStPodigrati As Integer
 Dim intSSS As Integer
 Dim A As Integer

    'zaokroži hdcp za izračun števila podigranih udarcev
    intHdcp = Round(Hdcp, 0)

    'seštevek za prvih devet lukenj
    intPrvihDevet = 0
    intPrvihDevet = CLng(frmVnosRezultatov.txtScoreIN.Text)

    'seštevek za drugih devet lukenj
    intDrugihDevet = 0
    intDrugihDevet = CLng(frmVnosRezultatov.txtScoreOUT.Text)

    'brutto seštevek skupaj
    intBrutto = 0
    intBrutto = intPrvihDevet + intDrugihDevet

    'netto seštevek skupaj
    intNetto = 0
    intNetto = intBrutto - intHdcp

    intSSS = CLng(frmVnosRezultatov.txtSSS.Text)

    'izračun število podigranih udarcev
    intStPodigrati = (intSSS + intHdcp) - intBrutto

    Select Case intStPodigrati
       Case Is <= "-3"
```

```
            If Hdcp >= 28 Then Exit Function
            Hdcp = Hdcp + 0.1
        Case "-2" To "0"
            Hdcp = Hdcp
        Case Is > "0"
            A = intStPodigrati

            Do Until A = 0 Or Hdcp <= 28
                A = A - 1
                Hdcp = Hdcp - 1
            Loop

            Do Until A = 0 Or Hdcp <= 20.5
                A = A - 1
                Hdcp = Hdcp - 0.4
            Loop

            Do Until A = 0 Or Hdcp <= 12.5
                A = A - 1
                Hdcp = Hdcp - 0.3
            Loop

            Do Until A = 0 Or Hdcp <= 5.5
                A = A - 1
                Hdcp = Hdcp - 0.2
            Loop

            Do Until A = 0 Or Hdcp <= -10
                A = A - 1
                Hdcp = Hdcp - 0.1
            Loop
            Hdcp = Round(Hdcp, 1)
    End Select

    IzracunHdcpSestevna = Hdcp

End Function
```

4. Programska koda za zapis novega igralca v bazo podatkov

```
Dim rsIgralec As New ADODB.Recordset

gSql = "Select Priimek, Ime, hdcp, StKlub, StSpol from t_igralec"

If rsIgralec.State = 1 Then rsIgralec.Close
rsIgralec.Open gSql, Povezava, 3, 2

If Not (rsIgralec.EOF And rsIgralec.BOF) Then
        Do
                If  Trim(UCase(rsIgralec!Priimek))  =  Trim(UCase(txtPriimek.Text))  And
                Trim(UCase(rsIgralec!Ime))       =       Trim(UCase(txtIme.Text))       And
                Trim(UCase(rsIgralec!Hdcp))      =      Trim(UCase(txtHDCP.Text))      And
                Trim(UCase(rsIgralec!Stklub)) = Trim(UCase(cmbStKlub.Text)) Then
                MsgBox "Ta igralec je že vnešen!", vbOKOnly, "Opozorilo!"
                Exit Sub
        End If
        rsIgralec.MoveNext
        Loop Until rsIgralec.EOF
End If

Povezava.BeginTrans
gTrans = True

gSql = "INSERT INTO T_Igralec (Priimek, Ime, HDCP, StKlub, stSpol)"
gSql = gSql & " Values('" & txtPriimek.Text & "','" & txtIme.Text & "','" & txtHDCP.Text &
"', " & cmbStKlub.Text & ", " & cmbStSpol.Text & ")"

Povezava.Execute gSql, stevilo

If stevilo = 1 Then
        Povezava.CommitTrans
        gTrans = False
        sbIgralec.Panels(1).Text = "Uspešno si vnesel podatke!"
        tmrIgralec.Enabled = True
Else
        Povezava.RollbackTrans
        gTrans = False
        sbIgralec.Panels(1).Text = "Napaka pri vnosu podatkov!"
```

```
        tmrIgralec.Enabled = True
End If

Call NapolniListBox
```

Najprej odpre nov zapis imenovan rsIgralec. Nato iz tabele igralec odpre vse potrebne spremenljivke. Računalnik odpre povezavo in preveri, če v bazi že obstaja zapis. V primeru da, preveri ali je igralec s tem priimkom, imenom, hednikepom in golf klubom že vnešen, če je uporabnika opozori in prekine vnos, v nasprotnem primeru nadaljuje z vnosom. Tukaj se začenja transakcija. V tabelo T_Igralec vnese posamezne vrednosti, ki jih prebere iz forme igralec. Ko računalnik izvede vnos v bazo preveri koliko zapisov je ta sprememba dodala. V primeru, da je to samo en, potrdi transakcijo in zapis imamo v bazi, uporabnik vidi na robu ekrana napis »Uspešno si vnesel podatke!«. V primeru da je zapisov vreč kot 1 ali celo noben, računalnik razveljavi vnos in uporabniku izpiše »Napaka pri vnosu podatkov!«. Na koncu se ta sprememba vidi v seznamu igralcev, ki se tako osveži.

5. Primer izpisa rezultatov za posameznega igralca

TURNIR

ORDER OF MERIT

Igralec: GREGORIČ UROŠ
Igrišče: BLED AB
Dan: 2 DAN

Datum: 17.5.2003
Način igre: SEŠTEVNA IGRA

Hole	Meter	PAR	Score	Hole	Meter	PAR	Score
1	351	4	4	10	135	3	3
2	538	5	5	11	416	4	6
3	169	3	4	12	461	5	5
4	360	4	5	13	177	3	3
5	453	5	5	14	330	4	5
6	307	4	4	15	365	4	4
7	150	3	3	16	497	5	4
8	295	4	4	17	485	5	4
9	381	4	4	18	386	4	4
OUT	**3252**	**36**	**38**	**IN**	**3004**	**37**	**38**
				TOTAL	**6256**	**73**	**76**

Stari HDCP: 2,1
Novi HDCP: 2,1
Fairway: 3
GIR: 12
Putts: 32
Chips: 4
Up and downs: 2
Sand saves: 2