UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

AGILNI RAZVOJ PROGRAMSKIH REŠITEV NA PRIMERU SISTEMA NAVISION

Ljubljana, januar 2006 MARKO SREBRE

IZJAVA

Študent Marko Srebre izjavljam, da sem avtor tega diplomskega dela, ki sem ga napisal pod mentorstvom dr. Tomaža Turka, in dovolim objavo diplomskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne 16. 1. 2006 Podpis: _____

Kazalo

1 Uvod

V današnjem času računalniško podprt informacijski sistem v podjetjih ni več izbira, ampak nuja. Velika količina podatkov ter iz njih pridobljene informacije so osnova odločanju podjetja. Podjetja, ki ne morejo hitro in pravilno reagirati na nenehne spremembe na trgu in v poslovnem okolju, niso konkurenčna. Dober informacijski sistem nudi informacije za pravilno in pravočasno odločanje, bodisi na podlagi trenutnih indikatorjev, bodisi na podlagi analiz preteklih podatkov. Toda vprašanje, ki se poraja že od začetka uporabe informacijske tehnologije, je, kako pravzaprav narediti dober računalniško podprt informacijski sistem. Pri razvoju informacijskega sistema namreč sodelujeta dve skupini ljudi z različnimi znanji in interesi. Na eni strani so tisti, ki so tehnično sposobni narediti sistem, na drugi pa tisti, ki razumejo poslovno področje, ki ga mora sistem podpirati.

V času so se pojavile različne metodologije razvoja informacijskih sistemov. Njihov namen je bil odpraviti prepad med kupci in dobavitelji sistema. Prve metodologije so se zanašale na jasno definirane postopke, ki vodijo delo razvijalcev sistema, tako da poskušajo analizirati čim večji del potreb kupca sistema in jih nato implementirati kot funkcionalnosti v sistem. Takšen pristop pa se ni izkazal za uspešnega pri vseh podjetjih. Vse bolj jasno je postajalo, da sledenje okornim postopkom in ustvarjanje velikih količin dokumentov, ki opisujejo delovanje sistema, ne more zagotoviti dobrega informacijskega sistema v poslovnem okolju, kjer se razni parametri nenehno spreminjajo.

Agilni razvoj, kot ime za filozofijo v ozadju številnih novejših metodologij, poskuša odpraviti dve težavi tradicionalnega razvoja sistema: neprilagodljivost in pretirano zanašanje na proces metodologije. Agilni razvoj je skozi nenehne iteracije prilagodljiv, poudarek pa ni na postopkih dela, temveč na ljudeh kot posameznikih. Tudi agilni razvoj ni univerzalna rešitev mnogih težav pri razvoju informacijskih sistemov. Pravzaprav deluje najbolje le tam, kjer je poslovno okolje turbulentno, specifikacije se nenehno spreminjajo, razvojne skupine pa so razmeroma majhne. V to skupino spada dejansko večina podjetij na prostem trgu, kjer se razmere nenehno spreminjajo, tako da je potrebno nenehno prilagajanje.

Navision je celovita poslovna rešitev za majhna in srednja podjetja, lahko pa zapišemo, da jo uspešno uporabljajo tudi v nekaterih večjih podjetjih. Je dinamična rešitev za dinamična podjetja na dinamičnem trgu. Navision ima svojo lastno metodologijo prodaje in implementacije. Izkušnja avtorja pa je takšna, da ta metodologija zadovoljivo deluje v fazi prodaje in analize sistema, nekoliko manj uspešna pa je pri oblikovanju, razvoju in implementaciji konkretnih specifičnih potreb kupcev Navisiona. Razvoj programskih dodelav ERP sistema Navision je tako v nevarnosti, da postane kaotična dejavnost.

Ideja te diplomske naloge je, da lahko uporabimo prvine in metode agilnega razvoja na primeru implementacije Navisiona. S sintezo agilnega razvoja in standardne Navision metodologije skušamo doseči boljšo zadovoljitev kupca glede njegovih specifičnih poslovnih potreb, skušamo doseči večjo prilagodljivost in boljšo upravičenost stroškov povezanih z izdelavo informacijskega sistema. Cilj diplomske naloge je tako podati prenovljeni proces

metodologije implementacije ERP sistema Navision, ki temelji na agilnem pristopu. Ta proces mora biti prilagodljiv, dinamičen in stroškovno upravičen.

Drugo poglavje opisuje osnovne pojme povezane z informacijskimi sistemi, pojasni, od kod potreba po metodologiji razvoja ter opredeli metodologijo življenjskega cikla razvoja sistema, ki služi kot primer procesno orientiranih metodologij.

Tretje poglavje predstavi agilni razvoj, manifest agilnega razvoja ter osnovne prvine, ki imajo pogosto ravno nasproten poudarek kot prvine metodologije življenjskega cikla razvoja sistema.

Četrto poglavje predstavi Navision kot poslovni sistem in kot razvojno okolje. Opredeli se metodologija Ontarget, ki je priporočena metodologija za implementacijo Navision rešitev.

Peto poglavje skuša združiti implementacijo Navisiona z agilnim razvojem, opisan je prenovljeni proces metodologije, podani so predlogi, kako med dobaviteljem in kupcem sistema oblikovati pogodbo, ki omogoča agilni razvoj.

2 Metodologije razvoja informacijskih sistemov

2.1 Vloga informacijskega sistema

V vseh organizacijah (podjetja, fakultete, cerkve, bolnice, knjižnice itd.) je prisoten nekakšen informacijski sistem. Tak sistem lahko opredelimo kot formalni ali neformalni (Gradišar, Resinovič, 1996, str. 109). Kot neformalni informacijski sistem razumemo tak sistem, ki ni predpisan; sem spadajo govorice, opravljanja, ideje in preference. Kljub nezanemarljivem vplivu takšnih neformalnih sistemov, nas ti v tem diplomskem delu ne zanimajo. Tema raziskovanja so formalni sistemi, to so takšni, ki ponujajo informacijo redno po vnaprej definiranem, predpisanem postopku.

> *"Informacijski sistem je sistem, ki zbira, shranjuje, obdeluje in ponuja informacije, ki so relevantne za organizacijo (oz. družbo) na tak način, da je informacija dostopna in uporabna tistim, ki jo želijo uporabiti, vključujoč upravitelje, osebje, stranke in državljane. Informacijski sistem je sistem človeških aktivnosti, ki lahko ali pa tudi ne vsebuje računalniške sisteme." (Avison, Fitzgerald, 1996, str. 13)*

Ali krajše:

> *"Informacijski sistem je sistem, v katerem se generirajo, arhivirajo in pretakajo sporočila in informacije." (Gradišar, Resinovič, 1996, str. 92)*

Definicija torej zajema vse možne oblike informacijskih sistemov. Vendar je tema raziskovanja te naloge še nekoliko ožja in nas zanimajo le računalniško podprti informacijski sistemi.

Čeprav se že od nekdaj srečujemo z informacijskimi sistemi, so ti šele nedavno začeli izkoriščati prednosti, ki jih ponujajo računalniki. Pred začetkom uporabe računalnikov so bili sistemi večinoma upravljani ročno. Uporaba informacijske tehnologije pomeni razširitev tega procesa. Vendar je ta razširitev signifikantnega pomena, saj ogromna kvantitativna sprememba, ki so jo omogočili računalniki s surovo močjo procesiranja velikih količin podatkov praktično brez napake, pomeni kvalitativen preskok v dojemanju informacijskega sistema, takšnem, kot si ga predstavljamo danes. Uporaba računalnikov je pripomogla v situacijah, kjer (Avison, Fitzgerald, 1996, str. 9):

- so povečane obremenitve preobremenile ročni sistem,
- je primerno osebje drago in ga je težko najti,
- je sprememba v načinu dela,
- so pogoste napake.

V računalniško podprtem informacijskem sistemu se uporablja informacijska tehnologija za hitro in natančno procesiranje (v primerjavi z ročnim sistemom, ki je počasnejši in manj natančen) podatkov ter za oskrbovanje z informacijami, ki so celovite in ravno prav podrobne, da so uporabne za nek namen.

Osnovni podatki, ki se procesirajo v današnjih informacijskih sistemih, so v glavnem tekst, številke, slika in zvok. Računalniško podprt informacijski sistem se lahko uporabi za shranjevanje podatkov (baze podatkov) ali pretvorbo podatkov v uporabne informacije (sistemi za podporo poslovnemu odločanju in poslovno obveščanje). Informacijski sistem organizaciji pomaga analizirati poslovanje in njeno poslovno okolje. Tako preverja, ali organizacija dosega svoje cilje oz. je na pravi poti proti tem ciljem. Cilji organizacije so lahko dobičkonosnost, dolgoročen obstoj, širitev, večji tržni delež ter zadovoljstvo strank in zaposlenih. Informacijski sistem organizaciji pomaga izboljšati učinkovitost posameznih operacij, tako da z informacijami, ki jih priskrbi, izboljša odločitve povezane z upravljanem. Mnogokrat se omenja kot dejavnik konkurenčne prednosti, organizacija brez formalnega in računalniško podprtega informacijskega sistema pa velja za konkurenčno zaostalo (Avison, Taylor, 1997, str. 1).

2.2 Uporaba računalniške tehnologije in nastanek potrebe po metodologiji razvoja

Prvi računalniško podprti informacijski sistemi (v obdobju do leta 1960) so bili implementirani brez pomoči eksplicitne metodologije razvoja informacijskih sistemov. V začetku je bil poudarek na programiranju in poklic programerja je tistihmal bil še posebej cenjen. Programerji pa kljub visoki stopnji tehničnega znanja niso bili nujno tudi dobri pri

sporazumevanju z uporabniki. To je lahko pripeljalo do slabe zadovoljitve potreb uporabnikov, takšen informacijski sistem pa se je dostikrat izkazal za neprimernega.

Redki razvijalci so pri svojem delu uporabljali kakršnokoli metodologijo. Namesto tega so se zanašali na grobe ocene in izkušnje. Določiti datum dokončanja informacijskega sistema je bilo tvegano, takšna ocena pa je bila velikokrat precej nerealna. Razvijalci so bili preobremenjeni, njihovo delo je v veliki meri pomenilo tudi popravljanje in izboljševanje že obstoječih in operativnih delov informacijskega sistema. Ko je nastopila nova zahteva po spremembi ali dodelavi že obstoječega in delujočega sistema, je to potrebo uporabnik sistema sporočil razvijalcu. Zahteva je nastala bodisi zaradi nedelovanja obstoječega sistema, bodisi zaradi sprememb v organizaciji in njenem okolju. Pogosto pa so takšne spremembe informacijskega sistema imele nepričakovane posledice za kak drug del sistema, ki ga je bilo treba zato naknadno popraviti, takšen začaran krog pa je povzročal nejevoljo med razvijalci in uporabniki[1].

Z naraščajočo uporabo računalnikov v organizacijah so upravljavci zahtevali vse bolj primerne informacijske sisteme, posebej spričo visokih stroškov implementacije sistema. To je pripeljalo do treh pomembnih sprememb v razvoju informacijskih sistemov (Avison, Fitzgerald, 1996, str. 9).

- Vse bolj pomemben je postajal tisti del razvoja informacijskega sistema, ki je imel opraviti z analizo in načrtovanjem (dizajn). Poleg vloge programerja na pomembnosti pridobi vloga sistemskega analitika.
- Spoznanje, da je, ko organizacija raste v velikosti in kompleksnosti, namesto posameznih rešitev vse bolj primerna skupna, integrirana rešitev informacijskega sistema.
- Želja po uveljavljeni metodologiji razvoja informacijskega sistema.

2.3 Metodologija razvoja

Metodologijo lahko na splošno opredelimo kot:

> *"...skupek postopkov, tehnik, orodij in pripomočkov za dokumentiranje, ki koristijo razvijalcem sistema pri njihovem prizadevanju implementirati nov informacijski sistem. Metodologijo sestavljajo faze, ki so same sestavljene iz podfaz in ki vodijo razvijalce sistema pri izbiri tehnik, ki so primerne v vsaki fazi projekta, ter jim pomagajo načrtovati, upravljati, kontrolirati in ocenjevati projekte razvoja informacijskih sistemov." (Avison, Fitzgerald, 1996, str. 10).*

Avison in Fitzgerald naprej ugotavljata, da je metodologija še več kot le skupek njenih sestavin. Navadno je osnovana na nekem *filozofskem* pogledu, sicer se ne razlikuje od metode. Metodologije se lahko razlikujejo v posameznih tehnikah, ki jih priporočajo, o postopkih, ki jih predpisujejo ali v vsebini posamezne faze, toda včasih gre celo za bolj fundamentalne razlike. Nekatere metodologije poudarjajo človeške vidike razvoja

1 Tudi danes se marsikje in mnogokrat razvija po načelu *code and fix*, ki ima povsem enake rezultate.

informacijskih sistemov, druge so osredotočene bolj na znanstveni pristop, nekatere so pragmatične, nekatere skušajo avtomatizirati čim več dela pri razvoju projekta. Različne metodologije lahko sledijo različnim ciljem. Ti cilji so lahko na primer (Avison, Fitzgerald, 1996, str. 11):

- Natančno zabeležiti zahteve informacijskega sistema. Metodologija pomaga uporabnikom določiti njihove zahteve in funkcionalne potrebe. Razvijalci sistema preučujejo in analizirajo uporabniške zahteve.
- Določiti sistematično metodo razvoja, kjer se lahko napredek učinkovito meri in opazuje. Težave pri velikih projektih, ki niso izvedeni v rokih, lahko povzročijo velike stroške ali druge posledice za podjetje. Dobro načrtovane faze razvoja in vmesni cilji v metodologiji omogočijo učinkovitost tehnik načrtovanja projekta.
- Izdelava informacijskega sistema v okviru časovnega roka pri sprejemljivih stroških. Čas namenjen posameznim tehnikam naj bo omejen, sicer je mogoče iskanju popolnosti posvetiti ogromno časa (Kovačič, 1998, str. 90).
- Izdelava sistema, ki je dobro dokumentiran in ga je preprosto vzdrževati. Prihodnje modifikacije informacijskega sistema so lahko posledica sprememb v organizaciji in njenem okolju. Dobra dokumentacija omogoča izvedbo modifikacij sistema z najmanj vpliva na druge dele sistema.
- Ugotoviti spremembe, ki jim bo potrebno slediti čim bolj zgodaj v procesu razvoja sistema. S tem ko razvoj sistema napreduje skozi faze analize in načrtovanja do faze implementacije, se povečujejo stroški povezani s spremembami sistema. Prej ko realiziramo spremembe, manjši so stroški projekta.
- Izdelava sistema, ki ustreza ljudem, na katere sistem vpliva. Če sistem ustreza naročnikom sistema (stranke, managerji, uporabniki), je bolj verjetno, da se bo sistem uporabljal in bo bolj uspešen.

Našteti cilji predstavljajo le nekaj primerov ciljev, katerim lahko metodologija sledi. Metodologija se lahko osredotoča na več ciljev ali pa sledi zgolj enemu. Bistveno pri vsem tem pa je razumevanje, da v ozadju vsake metodologije stoji neka ideja, ki nas usmerja pri delu, tako da določi kvalitete informacijskega sistema, ki jih skušamo pri razvoju doseči.

2.4 Življenjski cikel razvoja sistema

V tem poglavju je opisana metodologija razvoja informacijskega sistema - življenjski cikel razvoja sistema, ki je prevladovala do leta 1980 in je v uporabi še danes. Ta metodologija je osnova mnogim kasnejšim metodologijam razvoja in kot taka predstavlja tradicionalni ali klasični model pri izvajanju projektov.

Proces v tej metodologiji poteka fazno, od rojstva ideje, njene uresničitve v obliki delujočega sistema in do trenutka, ko zaradi novih potreb sistem zastara in ga je potrebno nadgraditi ali pa nadomestiti z novim (Gradišar, Resinovič, 1996, str. 414).

Prve računalniške poslovne aplikacije so pokrivale le osnovne aktivnosti podjetij, kot so hramba datotek, izdelava poročil in dokumentov. Te aplikacije so vključevale osnovno procesiranje podatkov: kopiranje, pridobivanje, izpolnjevanje, sortiranje, preverjanje, analizo, kalkulacije in komunikacije. Ljudje, ki so implementirali sistem, niso nujno tudi dobro razumeli potrebe uporabnikov. Bili so predvsem izvedenci v tehnološkem pogledu. Pogosto so uporabniki težko sporočili svoje potrebe programerjem. Čedalje bolj očitna je bila potreba po novi vlogi v razvoju sistema, ki se tiče analize in načrtovanja. Potreba po analizi in načrtovanju je uvedla vloge sistemskega analitika in bolj podrobno delitev na poslovnega in tehničnega analitika.

Metodologija življenjskega cikla ima naslednje korake:

- študija izvedljivosti,
- raziskovanje sistema,
- analiza sistema,
- načrtovanje sistema,
- implementacija,
- pregled in vzdrževanje.

2.4.1 Študija izvedljivosti

Študija izvedljivosti preuči obstoječi informacijski sistem, ki je bodisi ročni bodisi računalniško avtomatiziran. Zahteve, katerim ustreza stari sistem, se pregleda in ugotovi nove zahteve, ki jih mora izpolnjevati novi sistem. Te zahteve morajo biti v okviru mej in omejitev sistema, posebej glede resursov, ki so na razpolago. Študija na kratko poda alternativne rešitve problema. Za vsako od rešitev navede tehnične, kadrovske, organizacijske in ekonomske stroške ter koristi razvoja in delovanja sistema. Vsak predlagani sistem mora biti izvedljiv (Avison, Fitzgerald, 1996, str. 21):

- legalno, kar pomeni, da je v skladu z nacionalnimi zakoni in, kadar je smiselno, tudi v skladu z mednarodnimi zakoni,
- organizacijsko in sociološko, kar pomeni, da je sprejemljiv za organizacijo in njeno osebje, posebej kadar zadeva pomembnejše spremembe v obstoječem procesu organizacije,
- tehnično, kar pomeni, da obstaja primerna tehnologija in znanje za razvoj informacijskega sistema,
- ekonomsko, kar pomeni, da je finančno dosegljiv in da so stroški upravičeni.

Med vsemi možnimi alternativami se izbere sistem z okvirnimi funkcionalnimi zahtevami. Predlog sistema se v obliki formalnega dokumenta ali poročila, ki ga spremlja predstavitev, poda managementu, ki predlog analitikov bodisi sprejme ali zavrne.

2.4.2 Raziskovanje sistema

Odobritvi predloga sistema sledi faza podrobnega raziskovanja aplikacijskega področja sistema, s katero se ugotovijo:

- funkcionalne zahteve obstoječega sistema ter v kolikšni meri ta sistem te zahteve izpolnjuje,
- funkcionalne zahteve novega sistema ter situacije in priložnosti, ki namigujejo na spremenjene zahteve,
- kakršnekoli omejitve,
- obseg in vrsta podatkov, ki jih je potrebno obdelati,
- mejni pogoji in odstopanja,
- problemi obstoječih metod dela.

Informacije zbrane v tej fazi bodo veliko bolj podrobne kot informacije zabeležene s študijo izvedljivosti. Pri raziskovanju sistema so v pomoč tehnike izpraševanja osebja, pisni vprašalniki, opazovanje aplikacijskega področja ter zbiranje obstoječih dokumentov in zapisov.

- Opazovanje nam daje vpogled v problemsko domeno, delovne razmere, ozka grla in metode dela.
- Intervjuji s posameznimi ključnimi uporabniki in skupinami uporabnikov so pomembna tehnika za preverjanje informacij. So tudi priložnost za spoznavanje z uporabniki, kar lahko zmanjša morebitni odpor do sprememb.
- Z vprašalniki se pridobijo informacije velikega števila uporabnikov sistema in tistih, na katere bo sistem kakorkoli vplival. Primerni so tudi za pridobivanje informacij iz oddaljenih lokacij.
- Pregled zapisov in dokumentov prejšnjega sistema lahko osvetli nekatere probleme, vendar mora analitik upoštevati morebitno zastarelost dokumentov.
- S pomočjo statistike se lahko na raznih podatkih uporabijo tehnike vzorčenja.

V fazi raziskovanja sistem se uporabi dokumentacija:

- Diagrami poteka pomagajo analitikom slediti toku dokumentov znotraj oddelkov. V diagramu poteka lahko nakažemo tudi procese, ki spremljajo nastanek posamezne dokumente (npr. preverjanje in odprava napak). Diagram poteka lahko prikazuje tudi višjenivojski pogled na potek posameznih funkcij, ki se izvajajo po številnih oddelkih organizacije.
- Organizacijski načrt, ki kaže strukturo poročanja ljudi v podjetju oz. oddelku.
- Specifikacije ročno nastalih dokumentov, ki prikazujejo podrobnosti dokumentov ki se uporabljajo v neavtomatiziranem sistemu.
- Diagrami, ki prikazujejo interakcijo med različnimi komponentami sistema – ljudje, strojna oprema.

2.4.3 Analiza sistema

V prejšnjih fazi zbrana dejstva o obstoječem sistemu analiziramo s pomočjo vprašanj:

- Zakaj obstajajo določeni problemi v sedanjem sistemu?
- Zakaj se uporabljajo ravno te metode dela?
- Obstajajo kakšne alternativne metode?
- Kakšna rast obsega podatkov se pričakuje?

S tem, ko skušamo razumeti vse aspekte obstoječega sistema in njegovo obnašanje, nakažemo smer izboljšav v novem sistemu.

2.4.4 Načrtovanje sistema

Kljub temu, da se že v fazi študije izvedljivosti nakaže načrt novega sistema, se lahko analitiki odločijo drugače na podlagi novih dejstev zbranih v fazi raziskovanja in preučenih v fazi analize. Tipično bo nov sistem podoben prejšnjemu sistemu, vendar bo uporabljal več računalniške avtomatizacije (še posebej, če prejšnji informacijski sistem ni vključeval računalnikov). Odpravljeni bodo problemi starega sistema.

Dokumentacija o načrtovanju sistema bo vsebovala:

- vhodne in izhodne podatke informacijskega sistema,
- procese (od katerih so nekateri avtomatizirani, drugi pa ročni), ki pretvarjajo vhodne podatke v izhodne,
- struktura datotek in podatkov v sistemu,
- varnostno politiko sistema,
- načrt testiranja in implementacije.

Pri izdelavi načrta sistema se podrobno določijo vhodna ter izhodna oblika podatkov, oblike datotek.

2.4.5 Implementacija

Različne aktivnosti, ki privedejo do končne implementacije sistema so:

- programiranje in testiranje programske opreme,
- nakup nove strojne in programske opreme, ki je potrebna za delovanje novega sistema,
- kontrola kakovosti – vse ročne in avtomatizirane procedure se testirajo, da ustrezajo tako uporabnikom kot analitikom, uporabniki se morajo strinjati v novimi metodami dela,
- izdelava sistemske in uporabniške dokumentacije novega informacijskega sistema,
- uvajanje in izobraževanje uporabnikov.

Z implementacijo sistema se zberejo in preverijo pravi (produkcijski in ne testni) podatki, s katerimi se napolni baza novega sistema, preverijo se varnostni postopki. S tem je organizacija pripravljena na prehod na novi sistem in opustitev starega. Prehod na novi sistem lahko poteka na več načinov (Gradišar, Resinovič, 1996, str. 419):

- Vzporedno delovanje starega in novega sistema v času uvajanja. Uporabniki na določen čas uporabljajo tako stari kot novi informacijski sistem, dokler novi sistem ni popolnoma operativen in je zaupanje v novi sistem popolno. Pomanjkljivost tega pristopa je dvojno delo uporabnikov, ki morajo operirati z obema sistemoma hkrati.
- Neposreden prehod iz starega na nov sistem, ki ga je smiselno uporabiti, kadar starega sistema sploh ni, ali pa je vzporedno delo onemogočeno. Tak pristop je povezan tudi z največjim tveganjem.
- Postopni prehod na nov sistem, kjer se posamezni deli novega sistema (moduli) fazno implementirajo. Kadar uvajamo posamezni del sistema z namenom, da ga testiramo pred implementacijo celotnega sistema, to imenujemo pilotno uvajanje.

Testiranje sprejemljivosti se konča, ko so uporabniki gotovi, da nov sistem deluje zadovoljivo in je v stanju, ko lahko postane popolnoma operativen (gre "v živo").

2.4.6 Izvajanje in vzdrževanje

Končna faza nastopi, ko je nov sistem že v uporabi. Potrebna je podpora strokovnega osebja, ki skrbi za kontinuirano in učinkovito delovanje sistema s tehničnega in vsebinskega vidika. Običajno so potrebne dodatne prilagoditve, katerih vzrok so lahko spremembe v organizaciji in njenem okolju, tehnološki napredek ali dodelave sistema, h katerim se je zavezal implementator sistema za določeno obdobje po začetku delovanja novega sistema. Nekaj dela je povezanega tudi z odpravo napak nastalih po začetku delovanja sistema.

2.5 Namen metodologije življenjskega cikla

Pregled sistema zagotovi, da novi sistem ustreza specifikacijam danim v fazi študije izvedljivosti. Ugotovi se, ali so stroški in čas izdelave v okviru prej predvidenih in načrtovanih.

Gradišar in Resinovič nakazujeta, da so metodologije za razvoj informacijskih sistemov, ki izhajajo iz pristopa na osnovi življenjskega cikla razvoja sistema, primerne za gradnjo obsežnejših sistemov, ki jih nameravamo uporabljati dalj časa. Njun prikaz življenjskega cikla, čeprav v osnovi zelo podoben, se razlikuje nekoliko v razdelitvi na posamezne faze in v primerjavi s tradicionalnim zgoraj opisanim procesom bolj ustreza sodobnemu pojmovanju življenjskega cikla. Razvoj je v vseh fazah cikla podrobno načrtovan, izveden, nadzorovan in dokumentiran z namenom, da se zmanjša možnost napak. Razvoj takega sistema traja dolgo časa in predvideva sodelovanje večjega števila uporabnikov in informatikov. Štiri faze so

razdeljene še na podfaze in si sledijo v naslednjem zaporedju (Gradišar, Resinovič, 1996, str. 421):

- začetek,
  - študija izvedljivosti,
  - načrtovanje projekta,
- razvoj,
  - podrobna analiza zahtev,
  - zasnova notranje zgradbe sistema,
  - nabava in namestitev strojne opreme
  - programiranje
    - oblikovanje programske kode
    - testiranje programa,
    - dokumentiranje programa,
    - testiranje sistema,
    - dokumentiranje,
- uvajanje,
  - načrtovanje uvajanja,
  - urjenje uporabnikov,
  - prehod na nov sistem,
  - testiranje ustreznosti
  - spremljanje delovanja sistema po uvedbi,
- izvajanje in vzdrževanje,
  - podpora tekočemu delu,
  - vzdrževanje.

Pomembno je poudariti, da faze pri razvoju projekta ne potekajo povsem zaporedno – gre za iterativen proces, ki predvideva vračanje nazaj, kadar se ugotovijo napake ali pomanjkljivosti v fazi, ki je sicer že opravljena.

2.5.1 Slabosti in prednosti metodologije življenjskega cikla sistema

Življenjski cikel razvoja sistema nudi mnoge potencialne prednosti v primerjavi z ad-hoc pristopom, kjer se ne uporablja nobene metodologije. Uporaba standardov za dokumentiranje pripomore k natančnosti in celovitosti specifikacij za razvoj sistema ter zagotavlja primerno obveščenost uporabnikov.

Sledenje tej metodologiji prepreči ali zmanjša zamude rokov za izdelavo sistema. S tem, ko lahko tehnologi in uporabniki ob koncu vsake faze ocenijo napredek, se zmanjša ali povsem odstrani nepričakovana rast stroškov in poskrbi, da koristi sistema niso nižje od pričakovanih.

Z razdelitvijo razvoja v faze in podfaze daje konvencionalni pristop, skupaj z boljšo komunikacijo in tehnikami uvajanja, večjo kontrolo nad razvojem računalniške aplikacije in omogoča uporabo projektnega vodenja in razvojnih orodij.

Kritika metodologije življenjskega cikla izpostavlja nekatere potencialne slabosti. Poudariti je treba, da so nekatere od teh slabosti novejše metodologije, ki gradijo na temeljih konvencionalnega pristopa, odpravile, tako da vse naštete slabosti ne veljajo nujno tudi za vse današnje metodologije, ki izvirajo iz življenjskega cikla razvoja sistema (Avison, Fitzgerald, 1996, str. 30-34).

- Slabo pokrivanje potreb managementa. Namesto, da bi informacijski sistem sledil ciljem organizacije, je v uporabi v glavnem za operativne naloge na najnižjem nivoju. Kljub temu, da nekaj informacij doseže upravljavski vrh in srednji nivo upravljanja, so računalniki uporabljeni večinoma za rutinske in ponavljajoče se naloge.
- Neambiciozno načrtovanje sistema. Novi informacijski sistem zamenja obstoječi sistem (ki je lahko računalniško podprt ali ne) in je pogosto podoben sistemu, ki ga zamenjuje. Ta slabost je bila posebej očitna v primeru zamenjave ročnih sistemov z računalniškimi, do izraza pa pride tudi pri zamenjavi računalniškega sistema, kjer je premalo poudarka na izboljšavi logike poslovnih procesov.
- Modeli procesov niso stabilni. Poslovni procesi in organizacija sama se pogosto spreminjajo in so kot taki nestabilni. Model procesov mora biti pogosto spremenjen in napisan na novo in je zato tudi sam nestabilen.
- Načrt, ki je zasnovan na izhodnih informacijah sistema, je neprilagodljiv. Spremembe v zahtevah povzročijo velike spremembe v dizajnu sistema in povzročijo zamude rokov ter nezadovoljiv in neprimeren sistem.
- Nezadovoljstvo uporabnikov je povezano z mnogimi informacijskimi sistemi. Mnogokrat se pričakuje od uporabnikov, da se specifikacije podajo v zgodnji fazi razvoja, kjer ti še nimajo pravih informacij, da bi se strinjali o natančnih zahtevah sistema. Uporabniki niso seznanjeni s tehnologijo in težko prispevajo k razpravi o načrtovanju dobrega sistema.
- Težave z dokumentacijo. Pomanjkanje uporabniške dokumentacije.
- Pomanjkanje kontrole. Kljub metodološkemu pristopu, ki omogoča ocene potrebnega časa, ljudi in drugih resursov za razvoja sistema, so te ocene nenatančne zaradi kompleksnosti nekaterih faz in neizkušenosti ocenjevalcev.
- Nepopolni sistemi. Pogosto izjemni primeri in situacije niso diagnosticirani v fazi raziskovanja. Take izjeme že zgodaj povzročijo težave v delovanju sistema.
- Čakalna vrsta razvoja. Uporabniki lahko čakajo več mesecev ali celo let od študije izvedljivosti do implementacije sistema. V tem času pride do večjih sprememb v poslovanju. Nekoč aktualne funkcionalnosti sistema so ob izdobavi sistema nepotrebne.
- Stroški vzdrževanja. V poskusu, da bi ujeli predvidene roke izdelave sistema, je mnogo rešitev narejenih na hitro: namesto da bi si vzeli čas za dober dizajn, se zdi v dani situaciji bolj primerno popravljati slab dizajn. To je eden izmed faktorjev, ki so pripeljali do velikega problema, kako vzdrževati obstoječe sisteme. Resursi porabljeni za vzdrževanje

sistema imajo tako večjo prioriteto kot razvoj novih aplikacij, kar povzroča nadaljnje nezadovoljstvo uporabnikov.

3 Agilni razvoj informacijskih sistemov

Razvoj programske opreme in informacijskih sistemov je v svojih začetkih in ponekod še danes kaotična in nepredvidljiva dejavnost. Težnje k uporabi metodologije razvoja izhajajo ravno iz omenjenih pomanjkljivosti in prve metodologije so poskušale z izdelavo natanko določenega postopka, ki usmerja razvoj, povečati učinkovitost in predvidljivost. Proces je razdrobljen na manjše, bolj podrobne korake, velik poudarek je na načrtovanju in planiranju. Takšen pristop se zgleduje po ostalih inženirskih in gradbenih disciplinah, primer metodologije, ki temelji na njem – življenjski cikel razvoja sistema, pa je opisan v prejšnjem poglavju.

Takšen "inženirski" pristop pa ima tudi svoje slabosti. Vsaka aktivnost je podprta s številnimi dokumenti in formularji. Razvoj poleg primarnega dela vključuje tudi sledenje smernicam kompleksnih metodoloških okvirjev (Bajec, Krisper, 2002, str. 2).

Kot odgovor na "birokratske" metodologije se je v zadnjih letih izoblikovala nova skupina metodologij, najprej znanih kot "lahke" metodologije, danes pa je uveljavljen pojem *agilne* metodologije. Agilne metodologije skušajo doseči kompromis med razvojem programske opreme kot nenadzorovano dejavnostjo in dejavnostjo, pretirano nadzorovano s procesom metodologije. Kompromis se odraža v nekaj pomembnih spremembah nasproti težkim metodologijam, kjer je najbolj očitna ta, da je za dano nalogo običajno potrebno manj dokumentacije.

Martin Fowler meni, da obstajata dve ključni značilnosti agilnih metodologij, iz katerih izvirajo vse bistvene razlike:

- Agilne metodologije se raje prilagajajo kot predvidevajo. V klasičnem pristopu je večji del razvoja sistema načrtovan do podrobnosti za daljše časovno obdobje razvoja. To deluje, dokler se zahteve in okolje ne spremenijo. Njihova narava je torej taka, da se upirajo spremembam. Agilne metodologije pa, nasprotno, sprejemajo spremembe, jim prilagajajo proces razvoja in se tudi same lahko spreminjajo.
- Agilne metodologije so bolj usmerjene k ljudem[2] kot k procesom[3]. Ideja metodologij, ki so procesno usmerjene, je, da se definira proces, ki deluje ne glede na to, kdo ga izvaja. Agilne metodologije privzemajo, da proces ne more nadomestiti veščin razvojne skupine. Proces naj le podpira razvijalce pri njihovem delu (Fowler, 2003).

2 People-oriented (angl.).
3 Process-oriented (angl.).

3.1 Agilna zveza

Lahke metodologije so bile prvič združene pod isti okvir na srečanju v začetku leta 2001 v ZDA. Z namenom najti svoje skupne točke so se zbrali predstavniki t.i. "lahkih" metodologij in nekaj drugih izvedencev industrije. Poleg zastopnikov posameznih metodologij (Adaptive Software Development, XP, Scrum, Crystal, Feature-Driven Development, Dynamic System Development Method) so bili med udeleženci srečanja: zastopnika interesov izkušenih programerjev, sicer avtorja "Pragmatičnega programerja"[4], predstavnik perspektive testiranja programske opreme in predstavnik modelno usmerjenega razvoja informacijskega sistema (Cockburn, 2002, str. 215). Namen srečanja ni bil v poenotenju različnih metodologij v *enotno lahko metodologijo* (Unified Light Methodology – ULM), pač so se zbrani posvetili identifikaciji skupnih smernic novih metodologij. Izbrali so besedo *agilen*, ki najbolje opiše značilnosti novih pristopov, sebe pa poimenovali Agilna zveza (Agile Alliance).

V naslednjih nekaj mesecih so izoblikovali seznam skupnih vrednot in principov, ki naj bi razvijalcev informacijskih sistemov omogočale hitro ter učinkovito delo z visoko odzivnostjo na spremembe. Rezultat dela je manifest agilnega razvoja programske opreme, ki sestoji iz štirih osnovnih vrednot in dvanajstih bolj podrobnih izjav, konsistentnih z osnovnimi vrednotami.

> *"Odkrivamo boljše načine razvoja programske opreme in pri tem pomagamo drugim. V tej dejavnosti vrednotimo:*
> - *bolj posameznike ter interakcije med njimi kot procese in orodja,*
> - *bolj delujoč izdelek kot temeljito dokumentacijo,*
> - *sodelovanje s kupcem kot pogajanje na osnovi pogodbe,*
> - *bolj odzivnost na spremembe kot sledenje planu.*
>
> *Dasiravno je vrednost v postavkah na desni strani, vrednotimo bolj tiste na levi." (Manifesto for Agile Software Development, 2005)*[5]

Dejavniki na desni strani stavkov so elementi, na katerih gradi metodologija življenjskega cikla razvoja sistema in ostale t.i. "težke" metodologije. Ti dejavniki niso nepomembni s stališča agilnega razvoja; so koristni, a ne bistvenega pomena. V manifestu lahke metodologije postavijo nekoliko drugačen seznam prioritet. V nadaljevanju so podrobno predstavljene vse štiri osnove vrednote agilnega razvoja.

3.2 Posamezniki in interakcija nasproti procesom in orodjem

Pristop na osnovi življenjskega cikla razvoja sistema predpisuje bolj ali manj strogo določen proces, v katerem posamezniki nastopajo v točno določenih vlogah. Posamezniki v tem pogledu niso pomembni, pomembne so le vloge, ki jih opravljajo. Posameznik je

4 The Pragmatic Programmer, Andrew Hunt, David Thomas, 1999.
5 V prilogi je angleška verzija celotnega manifesta agilnega razvoja programske opreme.

nadomestljiv. Vloge so določene in v primeru, da nekdo zapusti svoje mesto, lahko na mesto njega v vsakem trenutku vstopi nekdo drug. Agilni razvoj temu nasprotuje in poudarja pomembnost posameznikov. Dober proces projekta ne reši pred neuspehom, kadar v moštvu ni dobrih igralcev, slab proces pa lahko onemogoči tudi moštvo najmočnejših igralcev. Skupina močnih igralcev lahko pričakuje neuspeh tudi v primeru, da ne zna delovati kot moštvo (Martin, 2003, str. 4).

Agilni razvoj daje prednost posameznikom, ki komunicirajo bolje, ti pa niso nujno tudi tisti, ki so tehnično najbolj sposobni. Prava orodja so pomemben faktor uspeha, ni pa zaželeno, da je na njih prevelik poudarek. Preobilje orodij je lahko prav tako slabo kot njih pomanjkanje.

V tem pogledu agilni razvoj, namesto da bi predpisoval okolje (procesi in orodja), v katerem naj delujejo posamezniki, poudarja *samoorganiziranost* skupine posameznikov. To pomeni, da je moštvu prepuščeno, da si samo izbere okolje, ki mu najbolj ustreza glede na značilnosti članov moštva in značilnosti naloge, ki se mora opraviti.

3.3 Delujoč izdelek proti temeljiti dokumentaciji

Ena glavnih pridobitev metodološkega pristopa k reševanju problemov razvoja programske opreme je proces dokumentiranja rešitve implementacije. Programska koda sama po sebi je težko berljiva, tehnična dokumentacija pa je uporabnikom malo jasna. Agilni razvoj priznava pomembnost dokumentacije in obenem opozarja na slabosti in neučinkovitosti, ki nastanejo ob pretiranem poudarku na dokumentiranju sistema.

Ključno vprašanje je tako, koliko dokumentacije je pravzaprav smiselno imeti. Obsežni dokumenti vzamejo ogromno časa za izdelavo in še več časa za redno osveževanje. Dokumentacija, ki ni v skladu s trenutnih stanjem razvoja informacijskega sistema, postane kup laži in tako signifikanten vir dezinformacij. Agilni razvoj predlaga dokumentacijo, ki je kratka in jedrnata. Kratka tako pomeni največ deset do dvajset strani, jedrnata pa, da vsebuje smernice razvoja in le visokonivojske strukture sistema, ki ga opisuje.

Agilni metode kot edino pravo in popolno dokumentacijo prepoznajo programsko kodo samo. Rek, da koda ne laže (angl. “the code does not lie”), pravi, da je koda, navkljub težji berljivosti, edini nedvoumni vir informacij o strukturi sistema. Poudarek je tako na berljivosti programske kode in jasni razdelitvi procesov.

Drugi vidik, ki nadomesti pisno dokumentiranje, je komunikacija med člani skupine, ki razvija informacijski sistem. Sodelujoči poznajo vedno spreminjajoči se načrt projekta. Najhitrejši ter najučinkovitejši način prenosa informacij pa je človeško medsebojno sporazumevanje. Agilni razvoj se zanaša na implicitno znanje.

3.4 Sodelovanje s kupcem proti pogodbenim pogajanjem

"Tretja vrednota opisuje razmerje med ljudmi, ki želijo, da se sistem zgradi, in tistimi, ki sistem gradijo. V pravilno oblikovanem agilnem razvoju pojmov *mi* in *oni* ne poznamo. Smo le *mi*." (Cockburn, 2002, str. 218).

Agilni razvoj poudarja, da je za uspeh projekta bistveno tesno sodelovanje s stranko. Kritizira pogled, pri katerem se informacijski sistem enači z dobrino, ki se jo kupi za fiksno določeno ceno. Opozarja na nevarnosti konvencionalnega pristopa, kjer se poda opis in zahteve sistema, na podlagi katerih naj skupina razvije tak sistem v danem času in za dano ceno.

Značilnost takega pristopa je, da se na začetku določijo funkcionalne zahteve in izdela plan razvoja na osnovi teh zahtev. Pogodba se sestavi, tako da vključuje plan in zahteve, izvajalca pa zavezuje, da bo v določenem času za določeno plačilo izdelal informacijski sistem, ki bo ustrezal specifikacijam.

Agilni razvoj v ospredje postavlja sodelovanje s kupcem, kjer je pogodbeno razmerje šele drugotnega pomena in pri izjemno tesni povezanosti kupca in izvajalca morda sploh odveč (Cockburn, 2002, str. 218). Sodelovanje zagotavlja izvajalcu reden tok povratnih informacij. Izvajalec se tako ne zanaša (le) na pogodbo in začetne zahteve, pač pa razvoj sistema sproti prilagaja dejanskim potrebam in zahtevam naročnika.

Robert C. Martin navaja primer uspešne "agilne" pogodbe, ki so jo v letu 1994 sklenili za obsežen, večleten projekt. Razvojna skupina je bila deležna le razmeroma majhnega mesečnega plačila. Večja izplačila so nastopila ob pomembnejših fazah projekta, ko so bili kupcu dostavljeni večji skupki funkcionalnosti. Ti skupki niso bili podrobno določeni v pogodbi. Namesto tega je bilo v pogodbi zapisano, da se izplačilo izvede za skupek funkcionalnosti, ko bo le-ta prestal naročnikov test sprejemljivosti. Podrobnosti testov niso bile določene s pogodbo (Martin, 2003, str. 5).

Pomembno za agilni pristop je, da naročnik nenehno spremlja razvoj sistema. Zaželene so pogoste iteracije, ki so lahko kratke tudi le teden dni. XP[6] celo predpisuje stalno fizično prisotnost predstavnika kupca v razvojni skupini.

3.5 Odzivnost na spremembe proti sledenju planu

Lastnosti plana sta lahko tako ročnost kot prilagodljivost. S prvo mislimo časovni okvir, v katerem plan velja (kratkoročni, dolgoročni, mesečni, letni plan), z drugo pa, do kakšne mere plan dopušča spremembe. Dolgoročni plan, ki načrtuje predaleč v prihodnost, je izpostavljen nevarnosti, da izgubi stik z resničnostjo. Več dejavnikov lahko povzroči, da plan, ki ni prilagodljiv, postane zastarel. Poslovno okolje se nenehno spreminja, kar posledično spremeni funkcionalne zahteve naročnika. Dolgi razvojni cikel lahko ustvari prevelik

6 Ena izmed najbolj priznanih in uspešnih agilnih metodologij - Extreme Programming.

razkorak - naročnik dobi sistem, ki sicer morda ustreza na začetku podanim zahtevah, ne ustreza pa več novim potrebam naročnika, ki se prikažejo v luči šele, ko se sistem dejansko začne uporabljati. Funkcionalne zahteve se bodo največkrat spremenile, ko naročniki vidijo sistem v delovanju.

Tudi kadar so zahteve dane in smo prepričani, da se ne bodo spremenile, le s težavo ocenimo, koliko časa bo trajal razvoj.

Strategija planiranja naj bo takšna, da se planira podrobno za kratko obdobje in zgolj površinsko za daljša obdobja. Različne agilne metodologije določajo različno dolga časovna obdobja, razvojne cikle, znotraj katerih poteka fiksno planiranje. Razlikujejo se tudi mehanizmi za soočanje s spremembami prioritet. Agilne metodologije planirajo le znotraj kratkega obdobja, ki se imenuje iteracijski cikel. Iteracijski cikli tipično niso daljši od treh mesecev (Crystal), medtem ko XP priporoča cikel dolg le eden do dva tedna.

3.6 Iteracijski cikel

Bistvena značilnosti agilnih metodologij, ki jim omogoča nenehno prilaganje in visoko odzivnost na spremembe, je iteracijski cikel. Bistvo razvoja na osnovi iteracijskega cikla je v tem, da se ob pogostih časovnih intervalih izdela popolnoma delujoča, testirana in integrirana verzija končnega sistema, ki vsebuje le del vseh funkcionalnih zahtev. Iteracijski cikel se ponavlja tako dolgo, dokler v celoti ni zadoščeno zahtevam, ki jih mora informacijski sistem izpolnjevati.

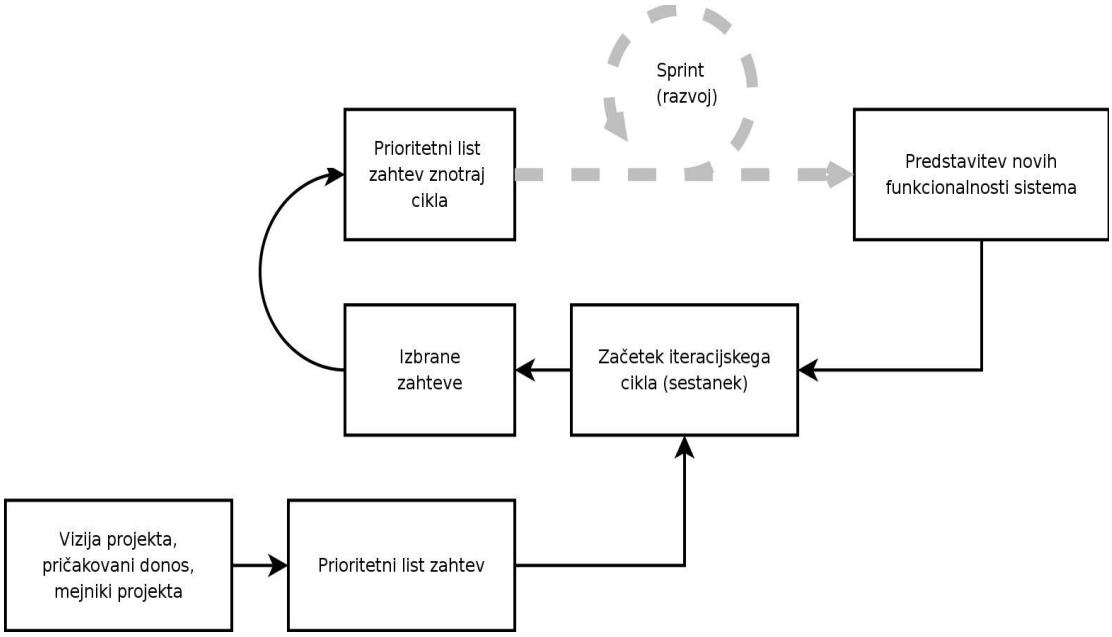
Časovni okvir enega cikla je razmeroma kratek in se med posameznimi agilnimi metodologijami razlikuje. Na koncu vsakega cikla dobi razvijalec povratne informacije od uporabnikov sistema, ki so bistvene za proces prilagajanja in odzivanja na spremembe. Posledica tega je, da nimamo več opravka s trdnim dolgoročnim planiranjem, edini gotovi plan je plan, ki je podan za tekoči iteracijski cikel. Vsi dolgoročni plani so izjemno spremenljivi in negotovi. Edini stabilen plan je kratkoročni plan znotraj enega iteracijskega cikla.

Nespremenljivost zahtev v okviru cikla daje razvijalcem možnost, da se osredotočijo na razvoj funkcionalnosti, ki so načrtovane v tem ciklu. Po drugi strani pa tudi v primeru, da se zahteve radikalno spremenijo, izgubljena vrednost ni večja od stroška enega cikla.

3.7 Primer agilne metodologije Scrum

Na Sliki 1 (str. 17) je prikazan proces metodologije Scrum. Osnova iteracijskemu ciklu je prioritetni list zahtev sistema (Product Backlog), ki izvira iz splošne vizije oz. cilja projekta. List zahtev se nenehno dopolnjuje in spreminja.

Slika 1: Agilni proces metodologije Scrum



Vir: Schwaber, Beedle, 2002, str. 8.

Vsak razvojni cikel se začne s sestankom razvojne skupine in skrbnikom prioritetnega lista zahtev. Skrbnik poda zahteve, ki so najvišje na prioritetni lestvici (npr. tiste, ki ponujajo največjo donosnost), moštvo pa poda oceno, koliko od teh zahtev je mogoče uresničiti znotraj naslednjega cikla. Sklene se dogovor, razvojna skupina izdela načrt za implementacijo. Sledi trideset dnevni iteracijski cikel (sprint), kjer se skupina usklajuje dnevno in na koncu izstavi delujoč, testiran in uporaben del sistema. V času trajanja iteracijskega cikla zahtev ni mogoče spreminjati.

Pri iteracijskem procesu metodologije Scrum delujejo tri vloge (Schwaber, 2004, str. 6):

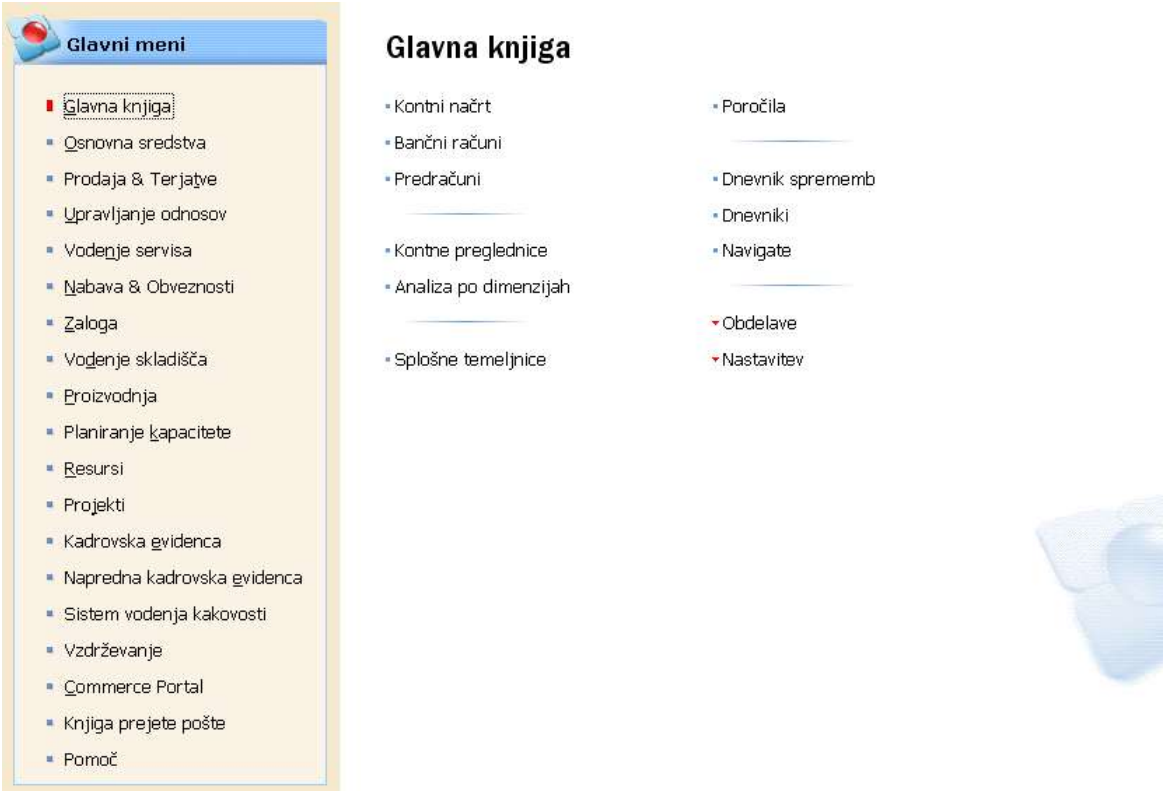
- **Pokrovitelj projekta** predstavlja interese naročnikov sistema. Njegova skrb je financiranje sistema in izdelava začetnih splošnih zahtev ter postavitev ciljev glede donosnosti. Pokrovitelj projekta skrbi, da so najbolj cenjene funkcionalnosti sistema najprej izgrajene.
- **Razvojna ekipa** je zadolžena za razvoj funkcionalnosti in se upravlja in organizirana sama; to pomeni, da člani sami izdelajo načrt, kako pretvoriti zahteve v nove funkcionalnosti sistema.
- **Scrummaster** je zadolžen za nemoteno in pravilo delovanje Scrum procesa, za prilagoditev procesa kulturi organizacije in izobraževanje udeležencev. Skrbi za to, da vsi sledijo pravilom in praksam agilnega procesa Scrum metodologije.

4 Uporaba agilnih praks pri razvoju informacijskega sistema v Navisionu

4.1 ERP sistem Navision

Microsoft Business Solutions Navision s polnim imenom ali na kratko Navision je programska oprema za upravljanje poslovnih aktivnosti, ki majhnim in srednje velikim podjetjem pomaga pri avtomatizaciji postopkov, pri sprejemanju bolj donosnih odločitev in pri pospeševanju rasti podjetja (Navision 4.0, 2005). Navision (od nedavnega Microsoftovega odkupa imenovani Microsoft Business Solutions - Navision) je celovita rešitev za podporo poslovnim procesom v podjetju.

Slika 2: Glavni meni grafičnega vmesnika Navisiona s seznamom modulov na levi



Vir: Lasten vir.

Dansko podjetje Navision A/S je že sredi osemdesetih let prejšnjega stoletja začelo z razvojem programske opreme za podporo poslovanju. V začetku se je Navision rešitev osredotočala na finančna in računovodska področja podjetja, sredi devetdesetih pa so se začele programski opremi s takratnim imenom Navision Financials dodajati funkcionalnosti, ki so pokrile tudi ostala področja poslovnega procesa. Kmalu je Navision prerasel v celovito rešitev za podporo poslovanja (ERP – Enterprise Resource Planning) z mnogimi funkcijskimi področji, ki jih ima danes. V juniju 2004 korporacija Microsoft kupi podjetje Navision in ustanovi poseben oddelek Microsoft Business Solutions, ki se ukvarja z razvojem celovitih

poslovnih rešitev. Število implementacij Navisiona po svetu znaša čez 130.000 (Valjavec, 2003, str. 31).

Zgradba Navisiona kot poslovnega sistema je modularna. Jedro rešitve predstavlja računovodski modul z glavno knjigo, s katerim so prepleteni praktično vsi ostali moduli. Vsak modul pokriva določen poslovni proces, izmed katerih si kupec rešitve lahko poljubno izbere tiste, ki jih potrebuje, upoštevajoč soodvisnost nekaterih modulov in nujnost osrednjega računovodskega modula. Posamezne gradnike lahko razdelimo v štiri skupine:

- Skupina modulov za računovodstvo in finance, ki predstavlja osrednje področje in jedro rešitve, v katero se povezujejo vsi ostali moduli.
- Skupina modulov za upravljanje z verigo dobaviteljev (Supply Chain Management - SCM) vključuje module za podporo procesov prodaje in nabave, skladiščenja izdelkov in surovin ter module za podporo različnim procesom v proizvodnji.
- Skupina modulov za upravljanje odnosov s kupci (Customer Relationship Management – CRM). Sem spadajo moduli za dodatno podporo modulom procesa prodaje in nabave. Namenjeni so zajemanju podatkov, ki se kasneje uporabijo za razne analize in podporo odločanju v podjetju.
- Skupino modulov za elektronsko poslovanje sestavljata modul Commerce Portal, ki je namenjen postavitvi spletnega portala za kupce, dobavitelje ter zaposlene, in modul Commerce Gateway, ki omogoča računalniško izmenjavo podatkov. (Valjavec, 2003, str. 31)

Dodatne module, ki niso del osnovne programske opreme, ponujajo različni, od Microsofta neodvisni dobavitelji. Posamezno podjetje, ki trži Navision, lahko izdela lastno modularno rešitev in jo trži naprej.

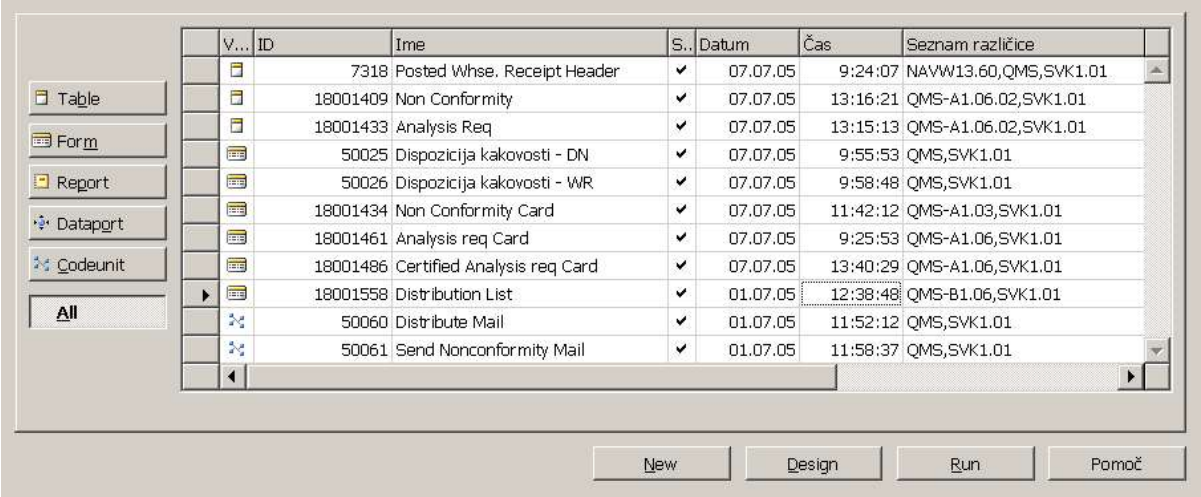
Navision programsko opremo trži svetovna mreža parternskih podjetij krovne organizacije Navision, ki sedaj spada pod oddelek Microsoft Business Solutions Navision. Takšen partner nosi ime Microsoft Certified Business Solutions Partner. Partnerji skrbijo za personalizirane lokalne storitve – od načrtovanja, implementacije do nadaljnje podpore – in zagotavljajo, da podjetje dobi natančno poslovno storitev, ki zadovolji vse njihove potrebe (Navision 4.0, 2005).

4.2 Navision kot integrirano razvojno okolje

Navision je, kot že omenjeno, ERP sistem, integrirana celovita rešitev, ki pokriva vsa področja poslovanja podjetja, vendar pa je pomembno, da razumemo, da je Navision hkrati tudi razvojno okolje za izdelavo te celovite rešitve. Navision v osnovni verziji ponuja neko, sicer zelo široko zastavljeno in modularno, a vendarle generično zasnovo informacijskega sistema. S številnimi moduli pokrije mnoga funkcijska področja, vendar praksa kaže, da ima vsako podjetje, v katero se sistem implementira, tudi svojo specifiko. Teh posebnih značilnosti je lahko malo, kot so manjše spremembe funkcionalnosti uporabniškega vmesnika

ali logike poslovnega procesa, ali pa veliko, kot na primer pri predelavi poslovne rešitve Navision v rešitev za podporo poslovanju v avtomobilski industriji (Incadea)[7]. Ne glede na obseg specifike pa pomenijo odstopanja od osnovne rešitve dodaten razvoj programske opreme v okolju Navision. Tako je Navision dvoje. Je že izdelan poslovni sistem za avtomatizacijo večine tipičnih poslovnih procesov, s katerimi se srečujemo v organizacijah.

Slika 3: C/SIDE razvojno okolje: seznam objektov v sistemu



Vir: Lasten vir.

Hkrati pa je tudi integrirano razvojno orodje (IDE – Integrated development environment) za razvoj informacijskega sistema, ki premore svoj lastni programski jezik (C/AL) ter strežnik/odjemalec arhitekturo za delo z bazami podatkov.

Navision baza podatkov lahko deluje na Navision strežniku ali na MS SQL strežniku.

V razvojno okolje vključujemo vsa programska orodja, s katerimi si pomagamo pri razvoju informacijskega sistema Navision: tekstovni urejevalnik kode, urejevalnik grafičnega vmesnika, urejevalnik definicij tabel, prevajalnik in razhroščevalnik programske kode.

S stališča diplomske naloge nas zanima Navision v obeh pogledih:

- kot celovita poslovna rešitev Navision v osnovni verziji pokriva večino področij poslovanja podjetja, kar vpliva na obseg nadaljnjega razvoja (je bistveno manjši kot, če bi bilo potrebno sistem razviti iz nič), ki je potreben za dosego posameznih potreb naročnika sistema,
- kot razvojno okolje, v katerem se izvede dodatni razvoj po zahtevah kupca.

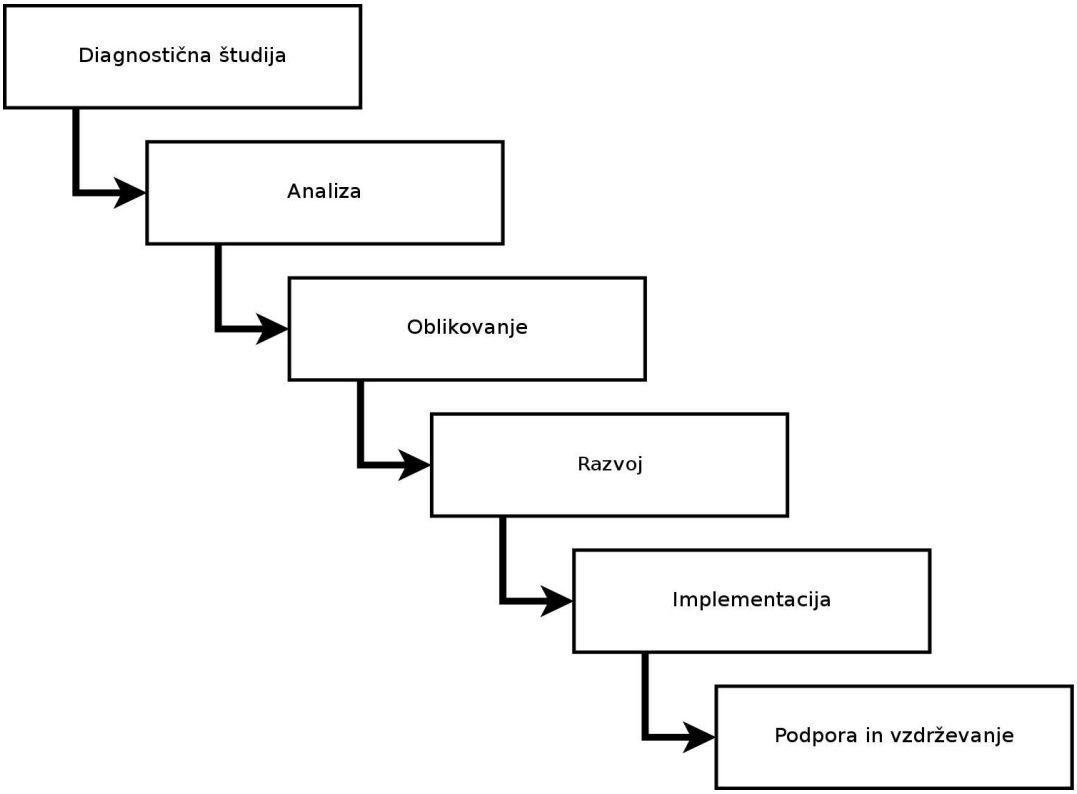
4.3 Metodologija Ontarget

Metodologija, ki se uporablja pri prodaji in implementaciji sistema Navision, se imenuje Navision Ontarget metodologija. To je priporočena in preizkušena metodologija, ki ne

7 Incadea.engine je predelana verzija Navision Financials, prilagojena za dejavnosti prodaje avtomobilov.

vključuje le same zasnove in implementacije programske opreme, ampak posameznih podjetjem, ki tržijo Navision (partnerjem), ponuja tudi smernice za strateško in letno načrtovanje ter analizo, segmentacijo trga, pozicioniranje in prodajo poslovne rešitve Navision Ontarget.

Slika 4: Navision metodologija implementacije



Vir: Lasten vir.

Navision metodologijo lahko širše razdelimo na dva sklopa (Slika 5, str. 22):

- predimplementacijska faza oz. *metodologija prodaje* in
- implementacijska faza oz. *metodologija implementacije*.

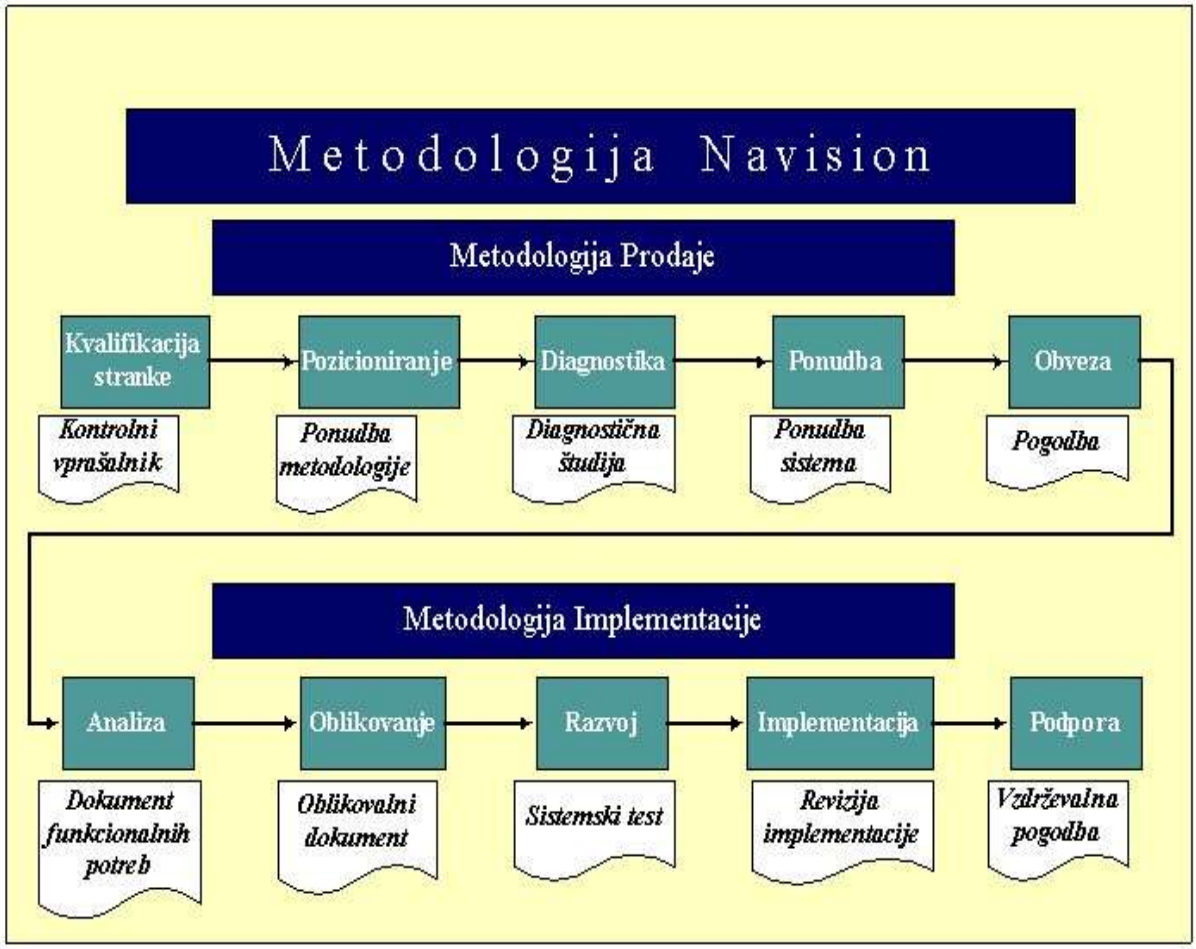
Oba sklopa sta razdeljena še na podfaze:

- metodologija prodaje,
  - kvalifikacija stranke,
  - pozicioniranje,
  - diagnostika,
  - ponudba sistema,
  - obveza,
- metodologija implementacije,
  - analiza,
  - oblikovanje,
  - razvoj,

- implementacija,
- podpora.

Ker so tema tega diplomskega dela metodologije razvoja informacijskega sistema, podrobnosti Ontarget metodologije prodaje niso predstavljene. Edina faza, ki nas zanima v tem kontekstu je faza diagnostike, ki jo lahko razumemo tudi kot fazo *študije izvedljivosti* življenjskega cikla razvoja sistema.

Slika 5: Faze v Navision metodologiji



Vir: Avtenta.si.

Posamezne faze implementacije so razdeljene na podfaze in te so naprej razdeljene na aktivnosti, podaktivnosti itn. V tabeli je prikazan primer uporabe projektnega plana, ki sledi zaporedju korakov procesa Navision metodologije. Primer je povzet iz generičnega projektnega plana podjetja Avtenta.si. Skupno število aktivnosti primera je 211, vendar je v tabeli v prid preglednosti prikazan le manjši del aktivnosti (celotna tabela aktivnosti projektnega plana se nahaja v prilogi). Aktivnosti si sledijo zaporedno ali vzporedno, v stolpcih so podani opis naloge, začetek, konec in trajanje posamezne aktivnosti.

Vsaka aktivnost ima določene tudi vloge, ki sodelujejo pri njej. Primeri vlog so: analitik, razvijalec, projektni vodja, Navision prodajalec na strani izvajalca ter glavni direktor, ključni

Tabela 1: Presek nekaterih faz in aktivnosti projektnega plana

Zap. št.	Naloga	Začetek	Konec	Trajanje
1	**Faza Analize**	**01/01/04**	**01/22/04**	**16d**
2	**Izdelava plana projekta**	**01/01/04**	**01/05/04**	**3d**
15	**Predstavitev in zagon projekta**	**01/06/04**	**01/06/04**	**0.5d**
17	**Priprava okolja pri naročniku**	**01/06/04**	**01/07/04**	**1.25d**
23	**Analiza poslovnih procesov**	**01/07/04**	**01/22/04**	**11.25d**
43	Predstavitev in primopredaja Projektnega plana in DFP naročniku	01/01/04	01/01/04	1d
44	**Osnovno izobraževanje KU**	**01/06/04**	**01/08/04**	**2.25d**
47	**Faza oblikovanja**	**01/23/04**	**02/11/04**	**13.25d**
48	**Priprava delavnic**	**01/23/04**	**01/23/04**	**0.5d**
51	**Izvedba delavnic**	**01/23/04**	**01/23/04**	**1d**
55	**Priprava oblikovnega dokumenta**	**01/23/04**	**01/27/04**	**3d**
58	...			
96	**Razvoj & Testiranje**	**02/11/04**	**03/15/04**	**23d**
97	**Priprava naročnikovega in Avtentinega razvojnega okolja**	**02/11/04**	**02/13/04**	**2.38d**
109	Zaključek razvojnega pristopa in projektnega upravljanja	02/11/04	02/12/04	1d
110	**Razvoj posameznih sklopov**	**02/18/04**	**03/15/04**	**18d**
123	Test na Avtentini bazi	02/11/04	02/12/04	1d
124	**Testiranje na bazi naročnika**	**02/11/04**	**02/18/04**	**5d**
136	Odobritev s strani naročnika	02/11/04	02/12/04	1d
137	...			
170	**Implementacija**	**03/15/04**	**03/19/04**	**4d**
171	**Priprava produkcijskega okolja**	**03/16/04**	**03/17/04**	**1.25d**
175	**Sprejetje testnih nastavitev**	**03/15/04**	**03/17/04**	**2.25d**
182	Prenos stanj in transakcij	03/15/04	03/16/04	1d
183	Pregled podatkov zaradi točnosti	03/15/04	03/16/04	1d
184	**Potrditev sistema (WalkThrough)**	**03/15/04**	**03/16/04**	**1.5d**
188	**Izobraževanje končnih uporabnikov s strani KU**	**03/15/04**	**03/19/04**	**4d**
193	Prehod v živo	03/15/04	03/16/04	1d
194	Potrditev prehoda v živo	03/15/04	03/16/04	1d
195	Potrditev projektnega in terminskega plana	03/15/04	03/15/04	0.25d
196	**Podpora in vzdrževanje**	**03/19/04**	**08/19/05**	**365d**
197	Seznanitev prodajalca NSC s stanjem na projektu	03/19/04	03/22/04	1d
198	**Priprava na delo po prehodu v živo**	**03/19/04**	**03/25/04**	**4d**
207	Po produkcijski sestanek za pregled dela na projektu	03/25/04	03/25/04	0.5d
208	Priprava in predstavitev podpore in vzdrževanja (Pogodbe)	03/19/04	03/22/04	1d
209	Izvajanje nadzora projekta	03/19/04	03/22/04	1d
210	Izvajanje podpore uporabnikom	03/19/04	08/19/05	365d
211	Dodatna šolanja	03/19/04	08/19/05	365d

Vir: Avtenta.si.

uporabniki, vodje oddelkov in vodja projekta na strani naročnika sistema. Vsaka od teh vlog pa ima po metodologiji predpisana potrebna znanja, ki jih mora imeti človek, ki to vlogo opravlja. Tako mora imeti analitik številna znanja kot so poznavanje področja industrije, izkušnje z izdelavami študij izvedljivosti, izkušnje z uporabo orodij, komunikacijske spretnosti, sistemsko znanje, predstavitvene sposobnosti, izkušnje z vodenjem delavnic, biti mora dober poslušalec (Navision Ontarget Toolkit, 2000).

V posameznih aktivnostih se uporabljajo predpisani obrazci, preglednice in predpripravljeni dokumenti, ki izvajalcem pomagajo pri ugotavljanju ključnih značilnosti sistema. Uporabljajo se orodja za modeliranje procesov, strukture sistema in vhodnih ter izhodnih vmesnikov. Navision Ontarget metodologija nudi vnaprej pripravljene dokumente, v katere izvajalec vnaša specifične podatke in jih v skladu z metodologijo oblikuje. Vsaka faza se zaključi s prenosom ključne dokumentacije, ki predstavlja rezultate te faze, v naslednjo fazo, kateri ta dokumentacija predstavlja vhodne parametre. Tudi rezultat posameznih aktivnosti je pogosto dokument ali diagram, ki predstavlja njene ugotovitve. Metodologija predlaga uporabo namenskega orodja Ontarget Modeler za modeliranje procesov, diagramov programske in strojne opreme, diagramov oddelkov in primerov uporabe.

4.4 Diagnostika

Diagnostika je faza, ki spada v prodajno metodologijo Ontarget, obenem pa pomeni tudi začetno fazo implementacijske metodologije. V tem času še navadno ni jasno, ali bo Navision partner dobil oz. sprejel projekt ali ne. Namen te faze je:

- s svetovanjem pridobiti konkurenčno prednost,
- zmanjšati tveganje s pridobitvijo soglasja glede vizije, ciljev in obsega projekta,
- razumeti in dokumentirati potencialna tveganja,
- določiti kompleksnost projekta,
- zbrati dovolj informacij za pripravo ponudbe sistema.
- določiti splošne zahteve projekta,
- identificirati razna problemska področja.

Metodologija kot del faze diagnostike predlaga izdelavo programskega prototipa, v primeru, da je to koristno in izvedljivo. Prototip ne zajema celotne aplikacije, ampak se osredotoča na ključna področja sistema, tipično tista, pri katerih je izvedljivost vprašljiva.

Rezultati diagnostike so uporabljeni pri pripravi ponudbe sistema, katero naročnik bodisi sprejme ali zavrne.

4.5 Analiza

Faza analize vsebuje skupek aktivnosti, ki natanko in celovito definirajo, katere poslovne procese naj programski informacijski sistem podpira. Namen faze je zajeti vse funkcionalne zahteve kupca v enoten dokument funkcijskih zahtev (Functional Requirements Document). Prav tako se izdela t.i. Gap-Fit analiza, ki prikaže, kateri poslovni procesi so že podprti z obstoječim (generičnim) Navision poslovnim sistemom in kateri še niso. Cilji faze so:

- določitev poslovnih procesov, sistemskih prilagoditev, vmesnikov in pretvorb podatkov, ki naj jih sistem podpira,
- zbiranje naročnikovih zahtev na vseh poslovnih področjih,

- dokumentiranje obstoječih poslovnih procesov in opis bodočih izboljšanih poslovnih procesov.

4.6 Oblikovanje

Namen faze oblikovanja je:

- izdelati podroben načrt sistemskih modifikacij, ki jih predvideva dokument funkcionalnih zahtev,
- izdelati podroben načrt bodočih poslovnih procesov na osnovi Gap-Fit preglednice,
- izdelati podroben načrt vmesnikov ter vhodnih in izhodnih točk sistema,
- izdelati strukturo podatkov, ki jih bo potrebno prenesti za začetek delovanja sistema.

Vsi opisi morajo biti dovolj podrobni, natančni in vsebinsko celoviti, tako da je moč na njih osnovi razviti programsko rešitev v sistemu. Programerji bi tako načeloma lahko brez dodatnih podatkov na osnovi oblikovnega dokumenta razvili vse dodatne funkcionalnosti, ki jih sistem zahteva.

Aktivnosti v tej fazi podajo bolj natančno oceno stroškov implementacije sistema.

4.7 Razvoj in testiranje

Faza razvoja pomeni realizacijo oblikovne faze. Načrt izdelave sistema, ki ga poda faza oblikovanja, služi razvijalcem in programerjem kot opis vseh funkcionalnosti, ki jih je potrebno izdelati. Vodja razvoja mora razdeliti načrt v manjše, bolj obvladljive enote. Posamezno enoto metodologija imenuje *gradnik*[8], razdelitev pa koristi v več pogledih:

- Težje in kompleksnejše delo se dokonča zgodaj v procesu razvoja. Tako lahko že kmalu identificiramo področja, ki imajo lahko kritičen vpliv na uspešnost projekta.
- Posamezne gradniki sistema služijo kot mejniki, ob katerih projektni vodja redno poroča naročniku sistema o napredku na projektu.
- Izgradnje omogočajo strukturirano testiranje.
- Naročnik prejme deloma dokončan sistem med samim razvojem, kar omogoča povratne informacije.

Metodologija opozarja, da je v fazi razvoja kontrola nad zahtevami naročnika kritična: uporabniki prvič vidijo udejanjenje svoje vizije sistema. Rezultat tega je, da si želijo več oz. nekaj drugega od začetno načrtovanega. Projektni vodja mora slediti vsem spremenjenim zahtevam in oceniti, v kolikšni meri te spremembe v dani fazi implementacije vplivajo na čas, proračun in obseg projekta.

8 Angl. *build*, v danem kontekstu: posamezen del večjega sistema.

Velik del resursov se porabi pri izdelavi dokumentacije in testiranju sistema. Metodologija opozarja na nevarnosti jemanja bližnjic. Premalo časa posvečenega planiranju, pripravi in izvajanju sistemskih testov pomeni večje tveganje za nastanek kasnejših programskih napak, rezultat tega pa je lahko neizpolnjevanje naročnikovih pričakovanj.

Ontarget opredeli razvoj kot *iterativen proces* v smislu odkrivanja napak in popravljanja le-teh. Cikel je sestavljen iz razvoja opreme in testiranja opreme, kateremu spet sledi razvoj, ki odpravi morebitne odkrite napake in nekonsistentnosti programske opreme.

Testiranje je sestavljeno iz testa posameznih enot, sistemskega testa, ki ga izvede Navision partner in naročnikovega testa posameznega gradnika. Ko naročnik odobri gradnik, se ta vgradi v produkcijski sistem, proces pa se nadaljuje, dokler niso vse izgradnje testirane in je na vrsti končni sistemski test.

4.8 Implementacija

V tej fazi so analiza, načrtovanje, razvoj in testiranja novega sistema končani. Obseg projekta se je verjetno nekoliko spremenil, napake so bile odkrite in odpravljene. Pripravi se test sprejemljivosti, ki temelji na funkcionalnih zahtevah in kritičnih faktorjih uspeha, določenih v načrtu projekta. Ko je opravljen test sprejemljivosti, začne baza obratovati v živo na produkcijskem sistemu.

Lahko se zgodi, da baza začne obratovati kljub temu, da ne zadovoljuje vseh kriterijev sprejemljivosti. To se zgodi zaradi vpliva eksternih dejavnikov, ki vplivajo na potek projekta. Primer takšnega odstopanja je, ko mora glavna knjiga (finančni računi) začeti delovati na novem sistemu ob začetku novega leta (1. januar) in ne šele kasneje. V tem primeru se predčasno lahko zažene del sistema, ki pokriva problematično funkcionalnost. Najverjetneje bo večina kriterijev sprejemljivosti v trenutku predčasnega začetka delovanja že izpolnjena, preostali pa bodo izpolnjeni kmalu po tem.

Tipične aktivnosti za pripravo začetka delovanja novega sistema so:

- namestitev in nastavitev produkcijskega sistema,
- priprava testov sprejemljivosti,
- prenos saldov in transakcij,
- pregled pravilnosti prenesenih podatkov,
- začetek delovanja sistema v živo.

4.9 Vzdrževanje in podpora

Naročnik in dobavitelj sistema skleneta pogodbo o vzdrževanju in podpori sistema. Vzdrževanje se lahko sklene za daljše časovno obdobje oz. se periodično podaljšuje. Aktivnosti v tej fazi vključujejo:

- svetovanje,
- opazovanje morebitnih napak in odpravljanje le-teh,
- opazovanje prihodnjih potreb, ki so posledica rasti podjetja in poslovnega razvoja,
- izboljšave za prihodnje poslovne priložnosti,
- podpora za programsko opremo.

5 Uporaba agilnih principov pri razvoju sistema Navision

V prejšnjih poglavjih so bile predstavljene smernice agilnega razvoja in uradna metodologija razvoja ERP Navision. V tem poglavju skušamo uporabiti filozofijo agilnega pristopa na primeru Navisiona, ter ugotoviti, katere agilne metode so primerne in koristne pri razvoju informacijskega sistema Navision in v katerih okoliščinah (obseg projekta, velikost projektne skupine itn.).

5.1 Velikost projekta

Najprej omenimo smiselnost uporabe agilnih metod glede na velikost projekta in projektne skupine. Fowler kot primer zgornjih omejitev velikosti projekta navaja skupine okoli 50-100 ljudi (metodologije FDD, XP, Scrum). Sicer je agilni razvoj namenjen predvsem manjšim skupinam in ni primeren za velike projekte. V področje manjših projektov spada tudi večina slovenskih implementacij Navisiona. Tipično delo poteka v projektnih skupinah do 10 ljudi, s tem, da delo ob istem času lahko poteka tudi na več projektih. S stališča velikosti projektnih skupin je agilni razvoj primeren za implementacije in razvoj programske opreme v okolju Navisiona.

Nadaljnje poglavje se osredotoča na takšne projekte in ugotavlja smiselnost uporabe agilnih metod nasproti metodologiji Ontarget, ki bolj ustreza slapovnemu modelu.

5.2 Praktični vidiki uporabe Ontarget metodologije

Osebne izkušnje z delom na številnih projektih kažejo na to, da se uradna metodologija Ontarget uporablja razmeroma malo in le na nekaterih delih projekta. Tako bo tipično izdelana diagnostična študija, analiza in oblikovna študija, vendar se proces izdelave ne bo posvečal vsem podrobnostim, ki jih predpisuje metodologija. Posamezni koraki so izpuščeni, vrstni red je mestoma drugačen. Metodologija se upošteva le ohlapno.

Precej značilno je, da se povsem nevede agilni koncepti že uporabljajo in to v smislu razvoja programske opreme, kjer ni neobičajno, da udeleženci na projektu delujejo po nekem neformalnem iteracijskem ciklu, ki pa nima nujno konstante periodike. V tem smislu je razvoj programske opreme v Navisionu lahko zelo podoben procesu, ki ga vpeljuje Scrum

metodologija. Dodatne funkcionalnosti čakajo v vrsti za implementacijo, s tem da so najbolj pomembne pri vrhu prioritetne lestvice. Navision partner se zaveže, da bo v nekem obdobju del teh zahtev izpolnil. Takšne zahteve niso bile nujno specificirane v začetnih fazah projekta (v fazi analize), in so navadno predmet pogajanj in pogajalske moči dobavitelja (Navision partnerja) in kupca.

Takšen neformalni pristop, ki deluje izven metodologije, se lahko kmalu spremeni v kaotični razvoj z vsemi slabostmi:

- zahteve se venomer spreminjajo,
- razvojni cikel se prekinja z novimi nalogami in zahtevami,
- razvoj poteka po načelu "kodiraj in popravljaj",
- razvoj ene funkcionalnosti povzroči nedelovanje druge funkcionalnosti.

Znotraj neformalnega iteracijskega cikla se dodatne zahteve načrtujejo in razvijejo. Pri tem sodelujejo svetovalci, analitiki in razvijalci. Vse te vloge so večkrat združene kar v eni in isti osebi. Delo ni nujno razdeljeno na posamezne odgovornosti, torej tiste, ki se posvetujejo s stranko o namenu in delovanju zahteve, tiste, ki pripravijo načrt programske dodelave in tiste, ki implementirajo rešitev kot končno programsko kodo. Pogosto je celo zaželeno, da imajo programerji stik s ključnimi uporabniki, saj se tako izogne marsikateri neskladnosti in nejasnosti pri izdelavi specifikacije dodelave in zagotovi, da končni izdelek ustreza pričakovanjem naročnika.

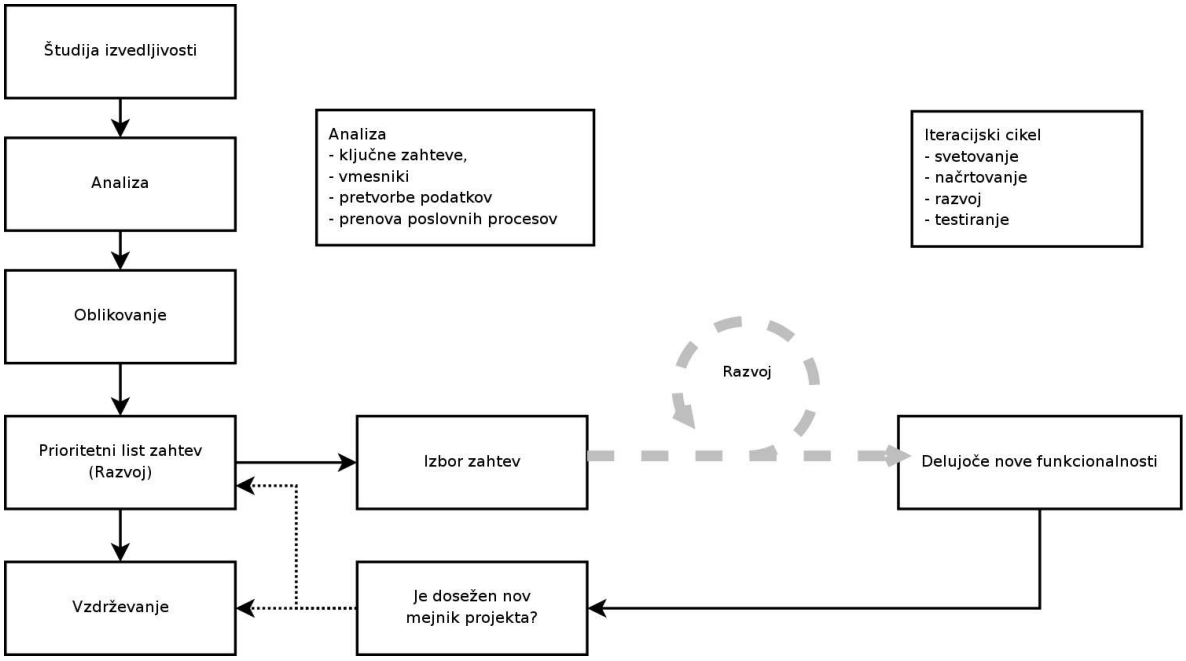
Takšen razvoj poteka docela neformalno in izven metodologije Ontarget. Čeprav bodo v pogodbi in planu projekta določeni mejniki in v oblikovanem dokumentu specificirane vse začetne zahteve, pa bo mnogo dodatnih zahtev postalo jasnih šele postopoma, ko se naročnik seznanja z delovanjem novega sistema, ko vidi prve delujoče inačice programske opreme in ko vidi rezultate prvotno zastavljenih zahtev.

Agilni koncepti tako v praksi na nek nepopoln, neformaliziran in kaotičen način že delujejo, ne da bi jih bilo potrebno imenovati z besedo *agilni*. V nadaljevanju pa nas zanima, ali lahko te agilne vidike formaliziramo in podamo predlog dopolnitve metodologije razvoja Navision sistema, ki bo vključeval iteracijski cikel in prvine oz. vrednote agilnega razvoja. Kar se tiče formalizacije procesa je vprašanje tudi, do kolikšne mere je smiselno formalizirati agilni proces.

5.3 Prenovljen proces metodologije razvoja Navisiona

Bistvena značilnost predloga prenovljenega procesa razvoja sistema je vpeljava iteracijskega cikla. Prvotni model, ki je zasnovan linearno in v katerem si faze sledijo zaporedno, razširimo v vpeljavo razvojnega cikla, ki je značilen za agilni proces (Slika 6, str. 29).

Slika 6: Prenovljeni proces



Vir: Lasten vir.

5.3.1 Začetek projekta

Vsak razvoj informacijskega sistema Navision se začne pravzaprav s prodajnimi aktivnostmi: pozicioniranje, izpostavljanje na trgu in lobiranje za projekte privedejo do prvih razprav o tem, kakšne so potrebe naročnika novega sistema, kaj lahko Navision partner ponudi in za kakšno ceno. Ta del se tiče agilnega razvoja le v tem, da se pravilno predstavi način dela in pridobi soglasje glede razmerja med naročnikom in dobaviteljem. To razmerje mora biti takšno, da je agilni razvoj možen.

Sledi faza diagnostike, kjer partner skuša oceniti kompleksnost projekta in pridobiti dovolj informacij za sestavo ponudbe sistema. V kolikor je ponudba uspešna, se na koncu te faze partner zaveže k izdelavi sistema. Ponudba vključuje licence za namestitev Navision strežnika in odjemalcev za uporabnike, granule, ki povedo, katere funkcionalnosti osnovnega sistema bodo v uporabi, ter dodatne zahteve, ki jih je potrebno še razviti. Za čimbolj natančno oceno investicije in stroškov lastništva sodelujejo v tej fazi prodajalci, svetovalci in analitiki. Z agilnega vidika je vsak razvoj nepredvidljiva dejavnost in njegovo mesto v pogodbi bo podrobneje predstavljeno v poglavju o agilni pogodbi.

Ključno vprašanje diagnostike in ponudbe sistema, ki sledi iz te faze je, kako sestaviti ponudbo sistema (pogodbo), ki bo upoštevala nepredvidljiv razvoj. Na ta vprašanje skušamo odgovoriti v poglavju Agilna pogodba.

5.3.2 Analiza

Faza analize v prenovljenem procesu ostane nespremenjena. Njen rezultat – Dokument funkcionalnih potreb – služi kot izhodišče za prioritetni list zahtev, na osnovi katerega se začne iteracijski razvoj. V agilnem razvoju pričakujemo kasnejše spreminjanje, dopolnjevanje in opuščanje funkcionalnih zahtev.

5.3.3 Oblikovanje

V fazi oblikovanja se oblikuje arhitektura in zamišljena implementacija sistema. Velik del oblikovanja, predvsem tisti, ki se tiče podrobnosti implementacije in razvoja, se odvije šele znotraj iteracijskega cikla. Oblikovalni dokument prenovljenega procesa naj zajame le ključne, visokonivojske strukture, podrobnosti implementacije pa ugotovijo svetovalci skupaj s stranko tekom iteracijskega razvoja.

5.3.4 Iteracijski razvoj

Na osnovi analize in oblikovalne študije se izdela prioritetni list zahtev. Svetovalci, ključni uporabniki in razvijalci se na začetku vsakega iteracijskega cikla odločijo, katere zahteve s seznama naj bodo implementirane v trajanju cikla. Priporočeno trajanje iteracijskega cikla je 2-3 tedne[9]. V stalnem sodelovanju s ključnimi uporabniki naročnika sistema se izdela načrt in poslovna logika implementacije, iz te pa sledi tudi programska rešitev problema. V tem pogledu začetni del iteracijskega cikla, ki načrtuje programsko rešitev, nadomesti oblikovalno fazo Ontarget metodologije implementacije. Bistvena razlika je, da se v prejšnjem procesu oblikovalna faza zgodi le enkrat, v prenovljenem procesu pa se zgodi v vsakem iteracijskem ciklu, za vsak sklop dodelav, ki jih je potrebno načrtovati.

Do konca iteracijskega cikla imajo razvijalci čas, da nemoteno razvijejo in testirajo predvidene rešitve. Ob koncu cikla je potrebno zagotoviti delujoče komponente programske rešitve, ki se lahko integrirajo v (produkcijski) informacijski sistem pri stranki. Naročnik ima tako možnost povratnega odziva in lahko glede na že delujoče komponente redefinira nadaljnje zahteve. Prednost iteracijskega razvoja je, da usklajevanje poteka sproti, ne šele na koncu razvoja, ko se dostavi celoten izdelek.

Prioritetni list zahtev se bo najverjetneje tekom razvoja spreminjal, nekatere zahteve se bodo opustile in dodajale se bodo nove. Naročnik ves čas prejema nove delujoče komponente sistema, začenja jih testno uporabljati, ob tem pa poteka še izobraževanje o novih funkcionalnostih sistema.

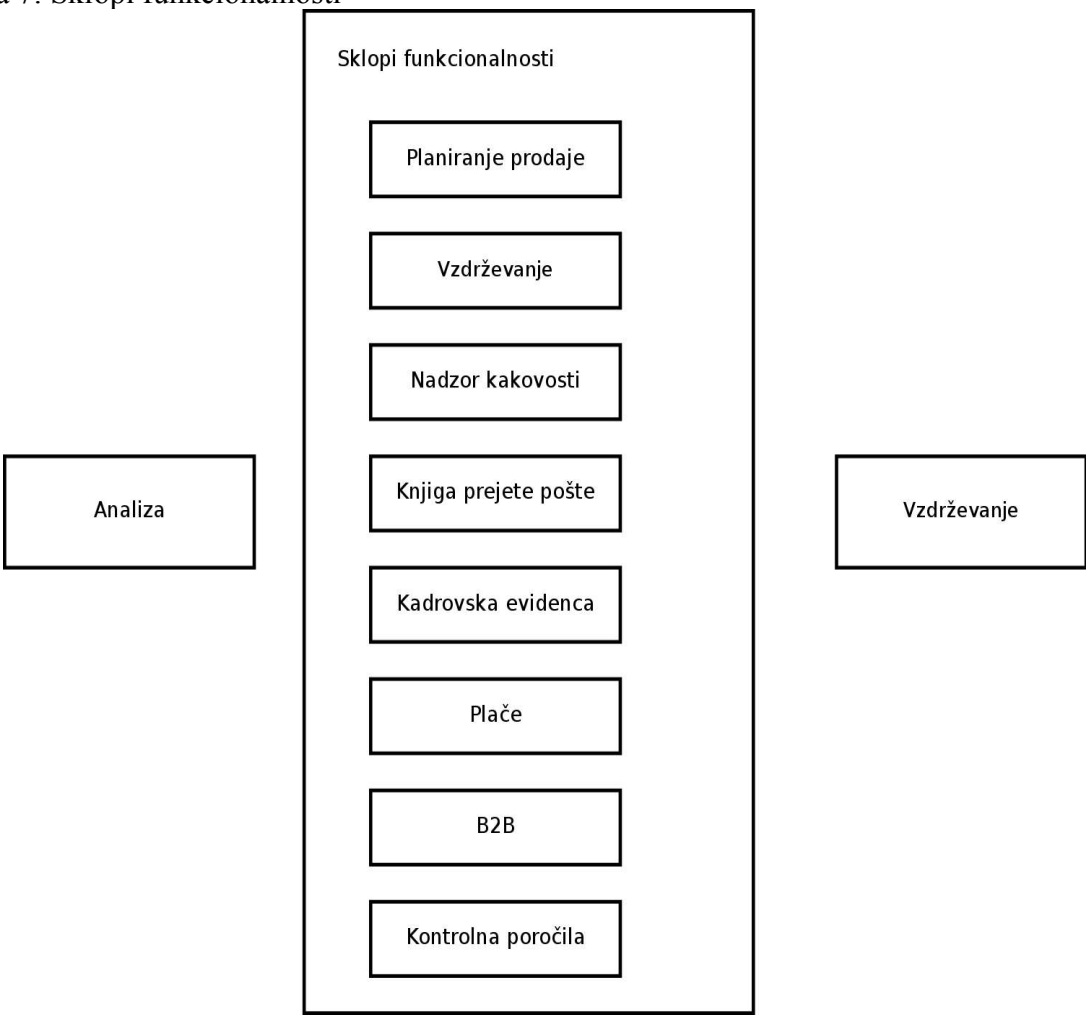
Razvoj večjih dodelav utegne preseči trajanje periode iteracijskega cikla. V takih primerih je potrebno dodelavo razdeliti na manjše obvladljive sklope, ki se lahko razvijajo znotraj cikla in ob končanem ciklu ponudijo del končne funkcionalnosti dodelave.

9 Odvisno od števila udeležencev in obsega ter kompleksnosti posamezih dodelav.

5.3.5 Mejniki in zaključek projekta

Zaradi nenehnega spreminjanja zahtev obstaja nevarnost, da projekt zaide izven svojega dosega. Načrtovanje je lahko težavno, težave nastanejo pri plačilu za opravljene storitve. Zato je smiselno, da se že v fazi analize določijo sklopi funkcionalnosti, ki so zgrajeni okoli kritičnih funkcionalnosti projekta. Takšni sklopi lahko služijo kot mejniki projekta. S tem določimo doseg projekta. Agilna pogodba ne more biti zasnovana na ključ. Večji zneski izplačil naj nastanejo ob predaji določenih sklopov funkcionalnosti. Naročnik in izvajalec se morata pri sestavi pogodbe in projektnega načrta strinjati, kateri sklopi funkcionalnosti morajo biti dostavljeni za uspešen zaključek projekta.

Slika 7: Sklopi funkcionalnosti



Vir: Lasten vir.

Na Sliki 7 (str. 31) je prikazan primer sklopov funkcionalnosti. To so kritične funkcionalnosti, ki so zahtevane za uspešno izpolnitev pogodbe. Primer je podan iz slovenskega proizvodnega podjetja s 500 zaposlenimi in 100 Navision uporabniki. Implementator je podjetje Avtenta.si. V agilno prenovljenem procesu se sklopi določijo na osnovi analize in so tudi del pogodbe.

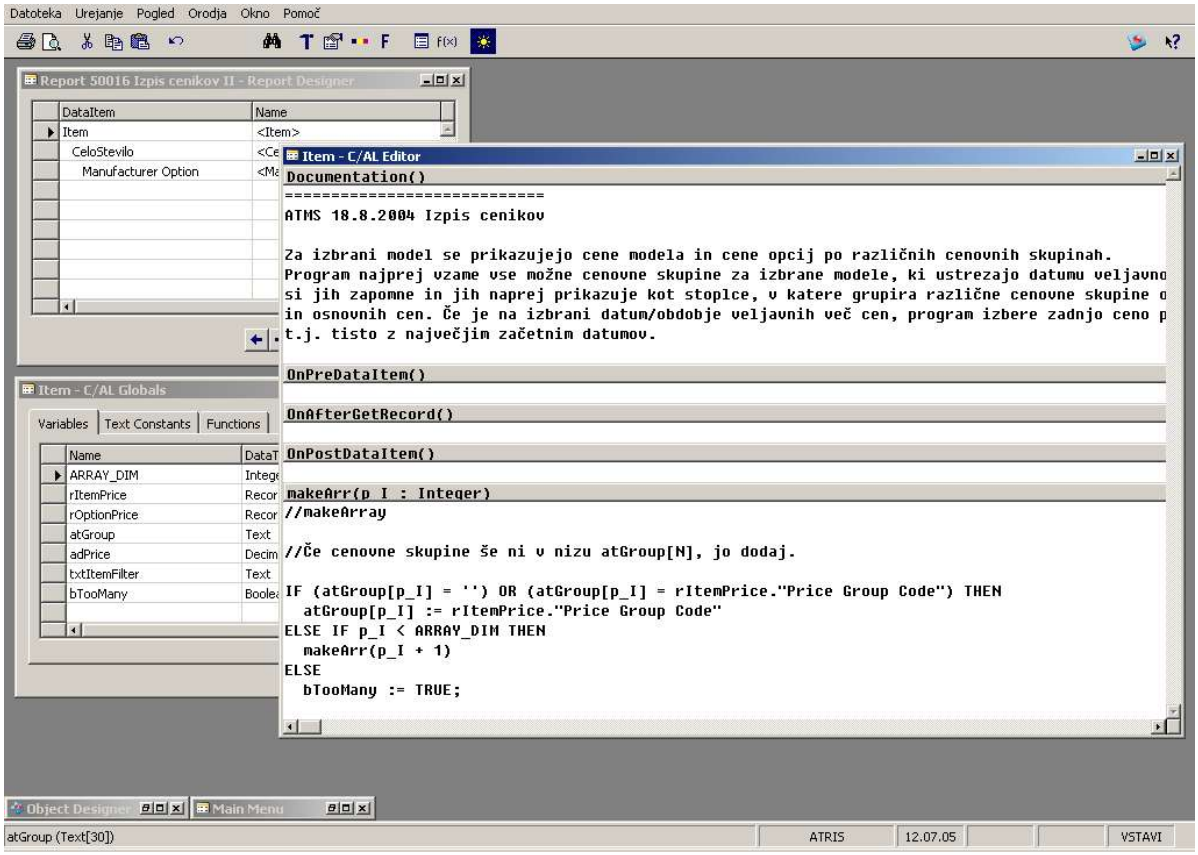
5.3.6 Vzdrževanje

S končanjem projekta nastopi faza vzdrževanja, kjer spet razvoj dodatnih zahtev deluje v okviru iteracijskega cikla, le da sedaj, odvisno od potreb naročnika, razvoj poteka le tako dolgo, dokler je to potrebno. Če naročnik v fazi vzdrževanja nima veliko dodatnih zahtev, bo prioritetni list zahtev v tej fazi večinoma prazen.

5.4 Dokumentacija

Dokumentacija v fazi analize v prenovljenem procesu ostane nespremenjena. Ugotovitve se dokumentirajo v dokumentu funkcionalnih potreb, ki služi kot osnova prioritetni listi zahtev.

Slika 8: Primer dokumentiranja programske kode znotraj Navision objekta



Vir: Lasten vir.

Oblikovalni dokument ne vsebuje več podrobnosti posameznih dodelav sistema. Izdela se načrt poslovnih procesov, načrt znanih vmesnikov z zunanjimi sistemi, določi se struktura podatkov, potrebnih za začetek delovanja sistema. Podrobnosti izdelave bodo ugotovljene v posameznem iteracijskem ciklu.

Kadar gre za dvofazno pogodbo, katere prvi del je vezan na fiksno ceno, je smiselno v fazi oblikovanja pripraviti tudi podroben načrt razvoja programske opreme.

V agilnem razvoju je bistven poudarek na dokumentaciji in preglednosti same programske kode. Glede na to, da orodja v razvojnem okolju Navision ne ponujajo veliko možnosti sledenja in dokumentiranja programske kode, je smiselno, da si organizacija sama postavi določena pravila, kako naj bo programska koda novih objektov in dodelav standardnih objektov dokumentirana.

Kupec in izvajalec sodelujeta pri oblikovanju uporabniške in tehnične dokumentacije. Slednja naj bo kratka in jedrnata. Podrobnosti arhitekture so znotraj programske kode, ki naj bo berljiva in ustreza standardom, ki jih postavi izvajalec.

5.5 Razvoj programske opreme

Razvoj poteka v iteracijskem ciklu, znotraj katerega si razvijalci sami organizirajo delo. Priporočljivo je tesno sodelovanje s ključnimi uporabniki naročnika sistema; v kolikor je možno, fizična prisotnost oz. dovolj velika možnost komunikacije (telefon, e-pošta, video konference).

Posamezne dodelave so združene v večje sklope funkcionalnosti, na osnovi katerih se lahko kupec in izvajalec dogovorita za izplačila. Sklopi služijo kot okvirni mejniki projekta.

Agilno moštvo se ne prilagaja le zahtevam kupca, ampak lahko prilagaja tudi svoj proces, ga preoblikuje (Fowler, 2000).

5.6 Agilna pogodba

Eden temeljnih problemov pri vpeljavi agilnega razvoja na osnovi iteracijskega cikla in prioritetnega lista zahtev je povezan s formalno opredelitvijo sporazuma med izvajalcem in kupcem.

Iteracijski model razvoja ponuja zgodnjo implementacijo uporabnih funkcionalnosti v poslovno okolje kupca. Uporabnik s sodelovanjem pri procesu zagotovi, da prejme poslovni sistem, ki ustreza njegovem namenu. Kupec nenehno sprejema odločitve glede funkcionalnih zahtev. Takšna opredelitev odgovornosti je nasprotje mnogih pogodb, kjer ima izvajalec pobudo, kupec pa le pasivno sprejema izdelek. To pripelje do situacij, kjer kupec želi vezati izvajalca na pogodbo s fiksno ceno. Takšen pristop je nepošten do izvajalca. Nasprotno želijo izvajalci pogodbo brez omejitev na čas in resurse. To pa je nepošteno do kupca (Stephens, 2001, str. 1).

V nadaljevanju je predstavljenih nekaj tipov pogodb, podano pa je tudi mnenje, zakaj so nekateri od teh tipov primerni za agilni razvoj, drugi pa ne.

5.6.1 Pogodba za dano ceno

Takšen tip pogodbe je najbolj pogosta oblika pogodbe, ki ščiti kupca. Proračunski cikli in povezani procesi v organizaciji lahko zahtevajo pogodbo za fiksno ceno. Za mnoge vladne organizacije zakon določa pogodbe za dano ceno, ki se navadno podelijo najboljšemu ponudniku.

Motivacija kupca v pogodbi za dano ceno je, da prenese tveganje na dobavitelja. Vendar je v končni fazi, če se pogodba ne obnese, prizadet tudi kupec.

Tipično bo pogodba za implementacijo sistema po Navision metodologiji Ontarget pogodba za fiksno ceno. Opravljenemu projektu pa sledi še vzdrževalna pogodba, ki je odprta in navadno vezana na mesečna pavšalna plačila.

Agilni razvoj je iteracijski razvoj; najbolj pomembne potrebe kupca se razvijejo najprej, razvoj pa se konča, ko zmanjka resursov. Takšen pristop je za dobavitelja skrajno tvegan in neprimeren v okolju s pogodbo za fiksno ceno. S stranko je težko doseči sporazum, da je delo opravljeno, ko zmanjka resursov. Zato agilni pristop ni primeren. Da bi se zaščitili, bodo dobavitelji raje oblikovali podrobne specifikacije sistema, jih striktno kontrolirali, ter zaračunavali posebej za vse dodatne spremembe. Rezultat tega je lahko, ali nezadovoljna stranka, ali bistveno povečanje stroškov implementacije (Beck, Cleal, 1999).

V okolju pogostih sprememb je prisotno veliko tveganje pri ocenjevanju stroškov implementacije. Kompetenten dobavitelj vključi to tveganje v ponudbo. Ponudnik, ki ne razume tveganja, bo najverjetneje stroške ocenil nižje. Proces izbiranja dobavitelja za pogodbo s fiksno ceno tako favorizira najbolj optimističnega oz. najbolj obupanega dobavitelja. Posledično je izbran dobavitelj, ki bo najverjetneje zašel v težave pri izpolnitvi pogodbe. Pogosto dobavitelj ne more izdobaviti izdelka oz. storitve za dano ceno. Dokler to ne postane očitno, je pogosto že prepozno za zamenjavo dobavitelja; kupec mora priskočiti na pomoč. Dobavitelj lahko poskuša poravnati svoje izgube skozi spremembe, kar pa privede kupca, da se agresivno izogiba vsem spremembam izven pogodbe.

Pogodba za dano ceno favorizira kupca in tveganje prenaša na dobavitelja, kar dobavitelja prisili v agresivno branjenje svojih interesov na račun kupca. Agilni razvoj, ki temelji na zaupanju in sodelovanju, v takšnem okolju ni možen oz. priporočljiv.

5.6.2 Pogodba za fleksibilno ceno (za čas in stroške)

Pogodba za fleksibilno ceno odpravi negotovosti in kompleksnost, vendar ne odpravi tveganja. Tveganje je prestavljeno iz dobavitelja na kupca. Dobavitelji imajo le malo iniciative delati učinkovito; dlje kot delo traja, več lahko zaslužijo. Zato so pogosto v veljavi dodatni kontrolni mehanizmi, ki pa predstavljajo dodatne transakcijske stroške, ki nimajo dodane vrednosti.

Težava takšne pogodbe je v tem, da je kupec, ko je enkrat sistem delno izgrajen, odvisen od dobavitelja, ta pa ima le malo iniciative, da zmanjša stroške. Agilni razvoj težavo delno odpravi s tem, da mora dobavitelj na koncu vsakega iteracijskega cikla dostaviti določeno vrednost za porabljen denar. Tako ima kupec možnost prekinitve pogodbe, ne da bi izgubil vrednost investicije v projekt do prekinitve (Poppendieck, Poppendieck, 2003, str. 169).

5.6.3 Delitev dobička oz. udeležba pri dobičku

Izvajalec se lahko s kupcem dogovori za udeležbo pri dobičku določenega izdelka, kateremu je namenjen informacijski sistem, ki se izdeluje. Obe strani sta visoko motivirani, da maksimizirajo uspeh (Sleumer et al., 2003, str. 2).

5.6.4 Pogodba s ciljnim stroškom

Pogodba s ciljnim stroškom je strukturirana, tako da so celotni stroški – vključno s spremembami – skupna odgovornost tako izvajalca kot kupca. Razlika od pogodbe s fiksno ceno je v tem, da, če je ciljna cena presežena, obe strani plačata več, kadar pa je celotni strošek pod ciljno ceno, si razdelita koristi. Dobavitelj ne pridobi dodatnega dobička, če dela dlje (za razliko od pogodbe na čas in stroške), lahko pa ima koristi, če konča prej ali pod ceno.

Pogodba se začne s splošnim sporazumom, kaj je potrebno doseči. Obe strani se morata strinjati, da podrobnosti v tej fazi niso znane in bodo šele, ko bo skupno delo končano. Sporazumeta se o končni ceni in časovnem poteku projekta. Ciljna cena, ki se določi, je bistvenega pomena: dizajn in načrtovane podrobnosti funkcionalnosti sistema bodo podrejene cilju, da se cena ne preseže. Določi se klavzula enakega deljenja stroškov, v primeru da bodo ti odstopali od ciljne cene (Poppendieck, Poppendieck, 2003, str. 171).

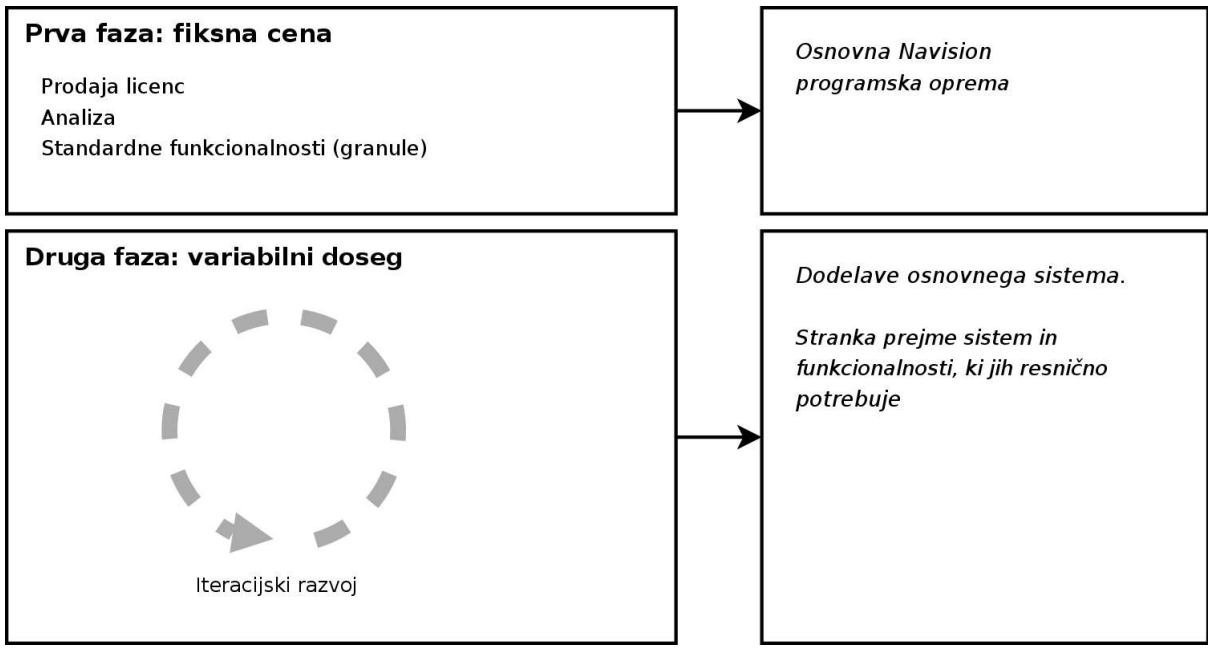
- Ciljni strošek ne vključuje dobička za dobavitelja. Plačilo, ki zagotovi dobavitelju dobiček, se izplača posebej. Navadno je izplačano po uspešnem zaključku del. Kadar skupni stroški presežejo ciljne stroške, si kupec in dobavitelj strošek razdelita. Kadar je skupni strošek pod ciljnim, prejme dobavitelj višje plačilo.
- Ciljni strošek vključuje dobiček dobavitelja, ki se strinja, da v primeru, da bo ciljni strošek presežen, zniža postavke in izključi dobiček.

5.6.5 Pogodba v dveh fazah

Pogodba je sestavljena iz dveh faz:

- V prvi fazi se izdela pogodba za fiksno ceno, vendar le za manjši del projekta. Dobavitelj sprejme preračunano tveganje, namen je, da se vzpostavi zaupanje med stranko in izvajalcem.
- Drugi del pogodbe je odprt in ni več vezan na fiksno ceno.

Slika 9: Dvofazna pogodba za agilni razvoj na primeru Navisiona



Vir: Lasten vir.

5.6.6 Variabilni doseg projekta

Skupna značilnost pogodb, ki so primerne za agilni razvoj je ta, da vse vsebujejo mehanizem, ki se izogiba fiksiranju dosega projekta v podrobnostih. Velik potencial povečanja produktivnosti pri razvoju programske opreme leži tudi v izpuščanju funkcionalnosti, ki niso potrebne (Poppendieck, Poppendieck, 2003, str. 176). Najboljši način, da se razvije kvalitetna programska oprema z manj stroški, je, da se piše manj programske kode. Sporazumi, ki omejujejo delo znotraj določenih stroškov in urnikov prisilijo razvojne skupine, da se začnejo spraševati, katere funkcionalnosti so resnično potrebne in katere se lahko izpustijo iz sistema.

Konvencionalni pristop temelji na natančni specifikaciji in kontroli dosega projekta, ki je določen v pogodbi. S tem naj bi se obe udeleženi strani, zavarovali ena pred drugo. Vendar je rezultat nizka in neoptimalna dodana vrednost. Kontrola dosega projekta navadno obseg razširi, ne pa zmanjša. To privede do povečanja stroškov implementacije funkcionalnosti in stroškov kontrole.

5.7 Pogodba za agilni razvoj na primeru Navisiona

Pogodba, ki izvira iz Ontarget metodologije je navadno pogodba za dano ceno in kot taka ni primerna za agilni razvoj. Pri prodaji in implementaciji Navision sistema je vsekakor potrebno upoštevati določeno specifiko Navisiona:

- gre za sistem, ki je deloma (glede na potrebe kupca) že izgrajen,
- licenca za uporabo in granule funkcionalnosti so fiksno določene v pogodbi.

Zato se pri implementaciji Navision sistema zdi smiseln dvofazni pristop (Slika 9, str. 36). Prva pogodba za fiksno ceno opredeli standardno Navision funkcionalnost, drugi del pogodbe pa je variabilen in odprt in se nanaša na dodatne zahteve, dodatne funkcionalnosti, ki jih potrebuje posamezni kupec. Drugi faza lahko sledi katerikoli pogodbi, ki je primerna za agilni razvoj:

- pogodba s ciljnim skupnim stroškom,
- pogodba na čas in stroške (s fiksno ceno iteracijskega cikla),
- udeležba pri dobičku.

Z razdelitvijo v dve fazi ohranimo del Ontarget metodologije, ki v prenovljenem procesu ustreza začetnim fazam opredelitve projekta (diagnostika, analiza, oblikovanje). Drugi del pa prikrojimo iteracijskemu razvoju, ki ga konvencionalna pogodba ne more zadovoljivo zajeti.

6 Sklep

Kadar ocenjujemo primernost posamezne metodologije razvoja informacijskega sistema, je vedno potrebno v razmislek vzeti tudi posamezne parametre projekta. Kakšen je obseg projekta, kako je določeno pogodbeno razmerje med kupcem in izvajalcem, koliko se bodo zahteve spreminjale, so le nekatera vprašanja, ki vplivajo na izbiro pravšnje metodologije. Pri implementaciji Navision poslovnega sistema je smiselno uporabiti agilni pristop, kadar imamo velik obseg modifikacij sistema, dodatnih programskih rešitev, ki jih je potrebno razviti, kadar zahteve niso jasno definirane, kadar je razvojna skupina relativno majhna. Dejansko so to dejavniki, ki so prisotni pri večini implementacij Navisiona, kadar gre za manjša in srednje velika podjetja. Iteracijski cikel razvoja ponuja nenehno prilagajanje in usklajevanje, vse to z namenom, da bi čim bolje razumeli in zadostili namenu informacijskega sistema, ki se razvija.

Največja težava agilnega pristopa utegne biti definicija pogodbenega razmerja. Za uspešno uporabo prvin agilnega razvoja je namreč v veliki meri potrebno ne le precejšnje sodelovanje kupca, temveč tudi njegovo razumevanje problematike ter zakaj in v kakšnih okoliščinah lahko izkoristimo prednosti iteracijskega razvoja. Verjetno je tako največji izziv, ki ostaja, pravilna in uspešna predstavitev procesa, ki temeljni na agilnih prvinah.

Literatura

1. Avison D.E., Fitzgerald G.: Information Systems Development: Methodologies, Techniques and Tools. London : McGraw-Hill, 1995. 505 str.
2. Avison D.E., Taylor V.: Information systems development methodologies: A classification according to problem situation. Journal of Information Technology, Hampshire, 12(1997), 1, str. 73-81.
3. Bajec M., Krisper M.: Agilne metodologije razvoja informacijskih sistemov. Uporabna informatika, Ljubljana, 11(2003), 2, str. 68-76.
4. Beck K., Cleal D.: Optional Scope Contracts. Eugene (OR) : Three Rivers Institute, 1999. 6 str.
5. Cockburn A.: Agile Software Development. Boston : Addison-Wesley Professional, 2002. 278 str.
6. Fowler M.: The New Methodology. Thoughtworks.
   [URL: http://www.martinfowler.com/articles/newMethodology.html], 2003.
7. Gradišar M., Resinovič G.: Informatika v poslovnem okolju. Ljubljana : Ekonomska fakulteta, 1996. 479 str.
8. Kovačič A.: Informatizacija poslovanja. Ljubljana : Ekonomska fakulteta, 1998. 214 str.
9. Martin Robert C.: Agile Software Development: Principles, Patterns and Practices. Upper Saddle River (N.J.) : Prentice Hall, 2003. 552 str.
10. Poppendieck M., Poppendieck T.: Lean Software Development. Boston : Addison-Wesley Professional, 2003. 240 str.
11. Stephens R.: Commentary on the draft DSDM contract. Ashford : DSDM Consortium. 11 str.
    [URL: http://www.poppendieck.com/pdfs/Commentary_on_draft_DSDM_contract.pdf], 2004.
12. Schwaber K., Beedle M.: Agile Software Development with Scrum. Upper Saddle River (N.J.) : Prentice Hall, 2002. 158 str.
13. Schwaber K.: Agile Project Management with Scrum. Redmond : Microsoft Press, 2004. 163 str.
14. Sleumer N., Arnoldi M., Milan M.: Pay per Use Contracts. Lifeware SA. 3 str.
    [URL: http://www.poppendieck.com/pdfs/Pay_Per_Use_Contracts.pdf], 2003.
15. Valjavec A.: Izmenjevanje e-dokumentov v nabavno prodajnem procesu med poslovnima rešitvama Microsoft Business Solutions–Navision in SAP R/3. Maribor : Fakulteta za organizacijske vede, 2003. 60 str.

Viri

1. Manifesto for Agile Software Development. Agile Alliance.
   [URL: http://www.agilemanifesto.org/], 10.8.2005.

2. Navision 4.0. Microsoft Business Solutions. Microsoft.
   [URL: http://www.microsoft.com/slovenija/businesssolutions/default.mspx], 5.6.2005.
3. Navision Ontarget Toolkit. Navision A/S, 2000.
4. Avtenta.si: Interna gradiva.

Priloga

Primer projektnega plana z vsemi podaktivnostmi Navision metodologije

Zap. št.	Opis naloge	Trajanje
1	**Faza Analize. št**	**16d**
2	**Izdelava plana projekta**	**3d**
3	Definicija vlog & Odgovornosti na projektu	0.25d
4	Definicija kritičnih faktorjev uspeha	0.25d
5	Definicija stopnje tveganja projekta	0.25d
6	Kreiranje ogrodja projektnega plana	0.25d
7	Definiranje plana dela	0.5d
8	Definiranje plana izobraževanja in podpore	1d
9	Definiranje načina komunikacije	0.25d
10	Vzpostavitev datumov za izvedbo delavnic za potrebe Analize	0.5d
11	Priprava datotek in izpisov za migracijo podatkov	0.5d
12	Priprava infrastrukture za izvedbo projekta	0.25d
13	Priprava Projektnega plana	1d
14	Zaključek Projektnega plana	
15	**Predstavitev in zagon projekta**	**0.5d**
16	Priprava prezentacije & predstavitev projektnim ekipam	0.5d
17	**Priprava okolja pri naročniku**	**1.25d**
18	Potrditev vsebine licence Navision	0.25d
19	Potrditev strežniškega in infrastrukturnega okolja	0.25d
20	Inštalacija klientov in podatkovne baze (SQL)	0.25d
21	Inštalacija Navision strežnika in lokalnih Cronus baz	0.25d
22	Priprava delovnih postaj za Super uporabnike	0.25d
23	**Analiza poslovnih procesov**	**11.25d**
24	**Priprava na intervjuje za izvedbo Analize**	**0.5d**
25	Priprava materialov za analizo poslovnih procesov	0.5d
26	Nastavitev in koordinacija poteka dela	0.5d
27	**Izvedba intervjujev**	**3d**
28	Izvedba Analize & Delavnic za pridobitev podatkov o poslovnih procesih	3d
29	**Pregled dokumentacije in ugotovitev**	**1.75d**
30	Pregled oddelkov, vlog in ciljev po oddelkih	0.25d
31	Analiza obstoječih in želenih procesov	0.25d
32	Dokumentiranje poslovnih procesov	1d
33	Identifikacija vmesnikov	0.25d
34	Identifikacija potreb po glavnih podatkih	0.25d
35	Identifikacija Dinamičnih/transakcijskih podatkovnih zahtev	0.25d
36	Definicija visoko zahtevnih funkcionalnih potreb	0.5d

Zap. št.	Opis naloge	Trajanje
37	**Priprava poročila analize in dokumenta funkcionalnih potreb (DFP)**	**6.5d**
38	Priprava analize delovnega toka	3d
39	Priprava GAP-fit analize	1d
40	Predlog izboljšanja poslovnih procesov (če je potrebno)	1d
41	Dopolnitev dokumenta s funkcionalnimi potrebami naročnika	1d
42	Priprava DFP	0.5d
43	Predstavitev in primopredaja Projektnega plana in DFP naročniku	1d
44	**Osnovno izobraževanje KU**	**2.25d**
45	Priprava izobraževalnega okolja (npr.: baze)	0.25d
46	Izvedba osnovnega izobraževanja ključnih uporabnikov	2d
47	**Faza oblikovanja**	**13.25d**
48	**Priprava delavnic**	**0.5d**
49	Priprava materialov za izvedbo delavnic	0.5d
50	Uskladitev in koordinacija datumov za izvedbo delavnic	0.5d
51	**Izvedba delavnic**	**1d**
52	Oblikovanje poslovnega modela	1d
53	Oblikovanje vmesnikov	1d
54	Oblikovanje konverzije podatkov	1d
55	**Priprava oblikovnega dokumenta**	**3d**
56	Priprava oblikovnega dokumenta	2d
57	Potrdilo oblikovnega dokumenta	1d
58	Pridobitev potrdila s strani nadzornega sveta za Oblikovni dokument-namenjen stranki	1d
59	**Oblikovanje razvoja po naročilu**	**8.5d**
60	Identifikacija vmesnikov po Analizi	0.5d
61	Oblikovanje tabel in podatkovnega modela	1d
62	Tok procesov	1d
63	Kreiranje rutin in funkcij	1d
64	Oblikovanje Form	1d
65	Spremembe okenskih prikazov	0.5d
66	Avtorizacije	0.5d
67	Revizija rezultata oblikovanja	0.5d
68	**Poročila**	**0.5d**
69	Oblikovanje & dokumentiranje poročil (našteti poročila)	0.5d
70	**Oblikovanje razvoja vmesnikov do zunanjih aplikacij**	**5d**
71	Identifikacija komponent za izdelavo vmesnikov	0.5d
72	Mapiranje vmesnikov	1d
73	Oblikovanje tabel in podatkovnega modela	1d
74	Tok procesov	1d
75	Diagrami vmesnikov	1d

Zap. št.	Opis naloge	Trajanje
76	Izdelava vmesnikov	0.5d
77	**Oblikovanje orodij za prenos podatkov**	**3d**
78	Identifikacija standardnih komponent za izvedbo prenosa podatkov	0.5d
79	Mapiranje podatkov	0.5d
80	Oblikovanje tabel in podatkovnega modela za pretvorbo podatkov	0.5d
81	Tok procesov za pretvorbo podatkov	0.5d
82	Diagram pretvorb	0.5d
83	Skupen pregled oblikovanj	0.5d
84	Šolanje uporabnikov za izvedbo prenosa podatkov	1d
85	Posodobitev razvojnih ocen	1d
86	**Potrditev detajlnega oblikovanja- Oblikovalni dokument za razvoj**	**4.75d**
87	Priprava Oblikovalnega dokumenta- za tehnično izvedbo	3.75d
88	Predstavitev Oblikovalnega dokumenta	0.5d
89	Pregled in razrešitev odprtih vprašanj glede Oblikovalnega dokumenta	0.5d
90	Podrobno oblikovanje zaključeno	
91	**Redefiniranje ponudbe in projektnega plana**	**1.5d**
92	Posodobitev Projektnega plana in konfiguracije licence	1d
93	Uskladitev stroškov projekta	0.5d
94	Priprava in predstavitev nadaljnjega plana implementacije	1d
95	Pridobitev odobritve s strani nadzornega sveta	1d
96	**Razvoj & Testiranje**	**23d**
97	**Priprava naročnikovega in Avtentinega razvojnega okolja**	**2.38d**
98	**Priprava naročnikovega okolja**	**2.38d**
99	Potrditev strojne opreme	1h
100	Inštalacija in konfiguracija Navision strežnika in baze	1d
101	Priprava testnega okolja	0.5d
102	Kreiranje produkcijskega okolja	0.5d
103	Inštalacija SW pri klientih naročnika- delovnih postajah	0.25d
104	**Priprava Avtentinega razvojnega okolja**	**2.13d**
105	Kreiranje baznega sistema, datotek in razvojnih standardov	2h
106	Kopiranje baznega sistema v razvojno	2h
107	Določitev postopkov za nadgradnjo objektov	2h
108	Kreiranje baze	4h
109	Zaključek razvojnega pristopa in projektnega upravljanja	1d
110	**Razvoj posameznih sklopov**	**18d**
111	**Razvoj sistema na osnovi plana**	**18d**
112	Priprava podatkov za testiranje	1d
113	Spremembe tabel in form	5d
114	Glavne funkcionalnosti in procesne rutine	5d
115	Migracija/pretvorba podatkov	2d
116	Vmesniki z drugimi aplikacijami	2d

Zap. št.	Opis naloge	Trajanje
117	Druge procesne rutine	2d
118	**Razvoj poročil**	**7d**
119	Pregled detajlnega oblikovanja poročil	1d
120	Razvoj poročil	5d
121	Testiranje poročil	1d
122	Dokončanje poročil	
123	Test na Avtentini bazi	1d
124	**Testiranje na bazi naročnika**	**5d**
125	**Zbiranje testnih podatkov podjetja**	**5d**
126	Zbiranje nastavitvenih podatkov	2d
127	Zbiranje glavnih podatkov (kupci dobavitelji artikli)	2d
128	Zbiranje prometnih podatkov (postavke)	5d
129	**Testiranje na testnih podatkih naročnika**	**1d**
130	Avtenta: Testiranje na testnih podatkih	1d
131	Posodobitev Navision Licenčne konfiguracije	1d
132	Vgradnja posodobitev pri naročniku	1d
133	Avtenta: Testiranje na testnih podatkih v okolju naročnika	1d
134	Naročnik: testiranje na testnih podatkih	1d
135	Testiranje zaključeno	1d
136	Odobritev s strani naročnika	1d
137	Implementacija v produkcijsko okolje	1d
138	Posodobitev razvojnega in testnega okolja pri naročniku	1d
139	Nastavitev uporabniških avtorizacij v produkcijskem okolju	1d
140	Postavitev splošnih nastavitev	1d
141	Nastavitev/migracija osnovnih poslovnih informacij	1d
142	Nastavitev/migracija poslovnih nastavitev	1d
143	Nastavitev/migracija Glavnih datotek	1d
144	Nastavitev/migracija poslovnih pravil	1d
145	Nastavitev/migracija odprtih postavk	1d
146	Migracija transakcijskih podatkov	1d
147	**Nastavitev in izvedba končnega sistemskega testa**	**3d**
148	**Izvedba končnega sistemskega testa**	**3d**
149	Potrditev naročnikovih podatkov (globalne nastavitve)	0.5d
150	Potrditev prenosa podatkov	0.5d
151	**Potrditev poslovnih procesov in delovnih tokov**	**3d**
152	Procesiranje dnevnih transakcij	1d
153	Procesiranje mesečnih transakcij	1d
154	Zagon dnevnih poročil in periodičnih poročil	1d
155	Potrditev delovanja zunanjih vmesnikov	1d
156	Zaključek testiranja	
157	**Pregled rezultatov testiranja**	**2d**

Zap. št.	Opis naloge	Trajanje
158	Pregled vprašanj/rezultatov- odprte zadeve	2d
159	Končna odobritev/zavrnitev testiranja (nadzorni svet)	
160	**Priprava na produkcijsko okolje**	**1.5d**
161	Pregled podatkov	0.25d
162	Terminiranje po-produkcijskih pregledov	1h
163	Definicija centrov pomoči, reševanja problemov in eskalacijskih procedur	1h
164	Zaključevanje šolanja za ključnih uporabnikov s strani Avtente	0.25d
165	Pregled uporabniške dokumentacije in priročnikov za šolanje	0.25d
166	Priprava Projektnega tima za prehod v živo	0.5d
167	Potrditev datuma prehoda v živo	1h
168	Sestanek nadzornega sveta za potrditev prehoda v živo	0.5d
169	Način upravljanje zahtev po spremembah v prihodnosti (po prehodu v živo)	1d
170	**Implementacija**	**4d**
171	**Priprava produkcijskega okolja**	**1.25d**
172	Nastavitev/naložitev produkcijskih podatkov	0.5d
173	Potrditev končnih in začetnih bilančnih stanj	0.5d
174	Prehod v živo & Odobritev	0.25d
175	**Sprejetje testnih nastavitev**	**2.25d**
176	Potrditev sprememb in zunanjih vmesnikov	0.25d
177	Backup obstoječega sistema	0.25d
178	Zaključek računovodskega obdobja na postavkah kjer je to mogoče	0.25d
179	Prenesi produkcijske podatke	0.5d
180	Potrdi transakcije (odprta stanja, vhodne postavke)	0.5d
181	Nastavi/naloži sprejeto podatkovno bazo	0.5d
182	Prenos stanj in transakcij	1d
183	Pregled podatkov zaradi točnosti	1d
184	**Potrditev sistema (WalkThrough)**	**1.5d**
185	Izvedba potrditve sistema (Walk Through)	1d
186	Pregled in potrditev prehoda v živo	0.5d
187	Sprejetje testa in nastavitev	
188	**Izobraževanje končnih uporabnikov s strani KU**	**4d**
189	**Skupina 1 uporabnikov**	**4d**
190	Skupina 1- šolanje v učilnici	2d
191	Skupina 1- šolanje v produkcijskem okolju	
192	Skupina 1 šolanje končano	2d
193	Prehod v živo	1d
194	Potrditev prehoda v živo	1d
195	Potrditev projektnega in terminskega plana	0.25d
196	**Podpora in vzdrževanje (opcijsko, po primopredaji projekta)**	**365d**

Zap. št.	Opis naloge	Trajanje
197	Seznanitev prodajalca NSC s stanjem na projektu	1d
198	**Priprava na delo po prehodu v živo**	**4d**
199	Pregled in potrditev poslovnih rezultatov	1d
200	Opazovanje dela sistema (performance)	1d
201	Opazovanje uporabe sistema	1d
202	Opazovanje podatkovne baze in verodostojnosti podatkov	1d
203	Opazovanje izkoriščenosti funkcionalnosti sistema	1d
204	Pregled začetnih Projektnih ciljev in namena	0.67d
205	Definicija po-implementacijskih vprašanj & reševanje	0.5d
206	Ocena naročnikovega zadovoljstva	0.33d
207	Po produkcijski sestanek za pregled dela na projektu	0.5d
208	Priprava in predstavitev podpore in vzdrževanja (Pogodbe)	1d
209	Izvajanje nadzora projekta	1d
210	Izvajanje podpore uporabnikom	365d
211	Dodatne šolanja	365d

Vir: Avtenta.si.

Manifest agilnega razvoja programske opreme

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	*James Grenning*	*Robert C. Martin*
Mike Beedle	*Jim Highsmith*	*Steve Mellor*
Arie van Bennekum	*Andrew Hunt*	*Ken Schwaber*
Alistair Cockburn	*Ron Jeffries*	*Jeff Sutherland*
Ward Cunningham	*Jon Kern*	*Dave Thomas*
Martin Fowler	*Brian Marick*	

Vir: Manifesto for Agile Software Development, 2005.