UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jernej Kržič

UVEDBA PROGRAMSKEGA MODELA WINDOWS WORKFLOW FOUNDATION V OBSTOJEČE APLIKACIJE
Na primeru rešitve AdLeasing

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor:
doc. dr. Mojca Ciglarič

Ljubljana, 2008

Zahvala

Zahvaljujem se mentorici doc. dr. Mojci Ciglarič za vodenje, strokovno pomoč in nasvete pri izdelavi diplomske naloge. Prav tako se zahvaljujem vodstvu podjetja Adacta d.o.o., da mi je omogočilo uporabo produkta AdLeasing kot pilotskega projekta za to nalogo. Svojim sodelavcem v podjetju pa se zahvaljujem za nekatere koristne strokovne napotke.

Kazalo

Seznam uporabljenih kratic

ASP.NET – Active Server Pages .NET

BPM – Business Process Management

GAC – Global Assembly Cache

IIS – Internet Information Services

JSON – JavaScript Object Notation

SOA – Service Oriented Architecture

WCF – Windows Communication Foundation

WF – Windows Workflow foundation

WPF – Windows Presentation Foundation

XAML – Extensible Application Markup Language

1 Povzetek

V diplomski nalogi sem obravnaval postopke uvedbe programskega modela Windows Workflow foundation v obstoječe aplikacije. Pri tem sem preučil različne pristope in podrobno opisal prednosti in omejitve vsakega izmed njih.

V začetku naloge sem opisal nekatere pomembnejše pojme, povezane s pojmom Upravljanje s poslovnimi procesi. Opisal sem programski model Windows Workflow foundation. Za lažje razumevanje pa sem predstavil tudi ostale tehnologije, ki so povezane z uporabo WF, kot so npr.: .NET Framework, Windows Communication Foundation, ASP.NET, itd. Praktični del naloge temelji na uvedbi modela WF v aplikacijo AdLeasing, ki sem jo tudi podrobno opisal.

Opisal sem cilje uvedbe modela WF v aplikacijo in obdelal posebnosti pri različnih tipih aplikacij, kot je ASP.NET. Posebno pozornost sem posvetil pravilnemu pristopu k vzdrževanju različic delovnih tokov, ki je po mojem mnenju prevečkrat spregledana tematika v literaturi, ki govori o delovanju WF.

V zadnjem poglavju sem predstavil postopke, ki sem jih izbral za uvedbo WF v obstoječo aplikacijo AdLeasing. V aplikaciji sem prepoznal dva poslovna procesa, kjer bodo možnosti WF prinesle največje koristi oz. največjo dodano vrednost. Predstavil sem tudi pomanjkljivosti v modelu WF, na katere sem naletel pri uporabi le tega.

Ključne besede:

BPM, Windows Workflow foundation, delovni tok, .NET Framework

1.1 Abstract

In this dissertation we discuss the process of implementing the Windows Workflow foundation model in existing applications. We present several possible approaches to solving this problem and their advantages, as well as limitations.

First, we describe the main concepts behind Business Process Management and the Windows Workflow foundation programming model. Other related technologies are also described, including .NET Framwork, Windows Communication Foundation, ASP.NET etc. The practical part of this dissertation focuses on introducing the WF model into an existing application – AdLeasing.

We present the goals of introducing the WF model and discuss specific issues, pertaining to its use with different types of applications, such as ASP.NET. It is the author's opinion that the problem of maintaining different versions of workflows is an issue often overlooked in existing literature so a significant part of the discussion focuses on this issue.

Finally, we present the procedures used to implement the WF programming model into an existing application. We focus on two business processes where using the WF model is likely to have the greatest benefit and also describe several shortcomings of the model, identified during this process.

Key words:

BPM, Windows Workflow foundation, Workflow, .NET Framework

2 Uvod

V diplomski nalogi sem najprej opisal osnovne pojme, povezane z upravljanjem s poslovnimi procesi (angleško Business Process Management oz. BPM). Skušal sem razjasniti nekaj zmede, ki jo prinaša uporaba kratice BPM za več različnih pomenov. Lotil sem se tudi problematike uvajanja BPM v podjetje. Prikazal sem življenjski cikel BPM in skušal pojasniti, kdaj je uvajanje BPM-a sploh smiselno in kakšni so ključni dejavniki, ki so potrebni za uspešno uvedbo. Opisal pa sem tudi, kakšna naj bi bila orodja za računalniško podporo uvedbe BPM.

V nalogi sem se usmeril predvsem na orodja in tehnologije, ki jih je za področje BPM pripravil svetovno znani proizvajalec programske opreme Microsoft. Da gre za pomembno področje, pa dokazuje serija produktov za podporo BPM tudi ostalih pomembnih proizvajalcev, kot so IBM, Oracle in ostali.

Namen te diplomske naloge je, opisati postopek uvedbe Microsoftovega programskega modela za podporo delovnim tokovom v obstoječo aplikacijo. Pri tem pa se nisem omejil le na aplikacije, razvite z najnovejšimi Microsoftovimi programskimi jeziki. Tako sem kot osnovo vzel sistem za prodajo leasing produktov AdLeasing, ki ga v podjetju Adacta razvijamo že več let. V njem sem prepoznal več poslovnih procesov, kjer bi lahko aplicirali prednosti delovnih tokov.

Kadar se soočimo z uvedbo novega pristopa v nekaj let staro aplikacijo, bomo pri tem zagotovo naleteli na vsaj nekaj možnih težav in razpotij. Glede na to, da je od maja 2002, ko je na trg prišla prva različica ogrodja .NET Frameworka 1.0, izšlo že pet dopolnjenih različic, se kaj hitro zgodi, da je aplikacija razvita za preveč zastarelo ogrodje. Žal prehodi med njimi pogosto niso trivialni. Prav tako moramo pri uvajanju podpore delovnim tokovom v spletne aplikacije vnaprej misliti na določene omejitve, ki so s tem povezane. Ker je WF le model oziroma ogrodje, nam ta omogoča več pristopov za uporabo le tega. Z izborom pravega se lahko na eleganten način izognemo prej omenjenim omejitvam. V nalogi sem opisal nekatere izmed teh pristopov ter opisal prednosti in slabosti vsakega izmed njih. Žal pa nisem mogel izmed njih izbrati samo enega, ki bi bil povsem univerzalen, ker vsak izmed njih prinaša drugačne prednosti in omejitve.

Po mojem mnenju je zelo pomembno poglavje, ki opisuje postopke za pravilno vzdrževanje različic delovnih tokov. Prej ali slej pridemo do potrebe po spremembi delovanja delovnega

toka, kar je lahko precej neprijetna izkušnja. Zato moramo posvetiti nekaj pozornosti pravilnemu pristopu k vzdrževanju različic. Zanimivo pa je, da temu poglavju nobena izmed knjig, ki sem jih preučil za pravilno razumevanje delovanja WF, ne namenja posebne pozornosti.

3 Predstavitev tematike in tehnologij

3.1 Upravljanje s poslovnimi procesi

3.1.1 Kaj je poslovni proces

Poslovni proces je seznam povezanih in strukturiranih aktivnosti. Njihovo izvajanje nas pelje proti končnemu cilju poslovanja.

Poslovni proces lahko pokriva delovanje zgolj ene poslovne enote združbe, ali pa se le ta razteza preko več enot. Čas izvajanja posameznega poslovnega procesa je lahko zelo različen. Poslovni proces lahko vsebuje tako avtomatizirane aktivnosti kot tudi ročne. Avtomatizirane aktivnosti so zmožne samostojnega računalniškega izvajanja znotraj izvajalnega okolja. Ročne pa tega niso zmožne, zato potrebujejo posredovanje izven meja izvajalnega okolja.

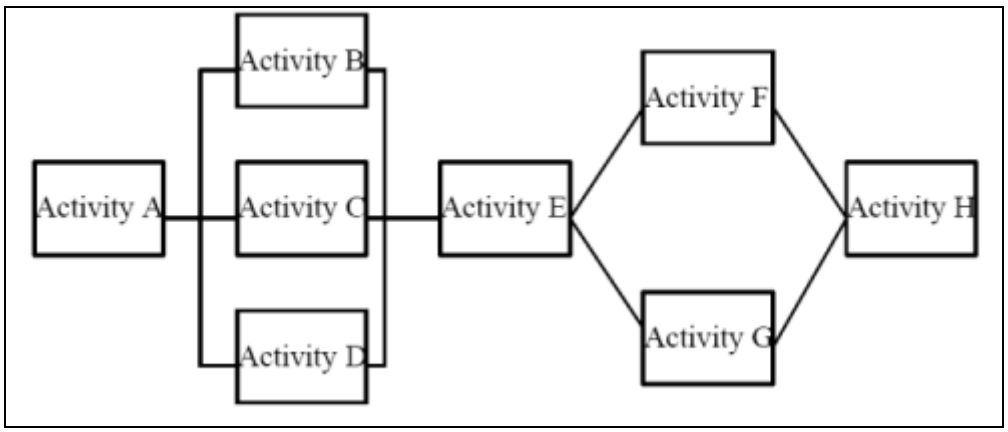
Poslovni procesi so definirani s tako imenovano procesno definicijo (ang. Process Definition), ki opisuje aktivnosti v procesu in poslovna pravila ter upravlja z uporabo podatkov med izvajanjem procesa.

Definicija poslovnega procesa lahko vsebuje tudi podprocese, ki so definirani ločeno in jih končni proces uporablja.

Primerek poslovnega procesa predstavlja posamezno izvedbo poslovnega procesa. Tako se primerek najprej ustvari na podlagi definicije poslovnega procesa, potem se izvaja in na koncu tudi zaključi. Pri tem je lahko vsak primerek povezan s svojim naborom podatkov, s katerimi se izvaja.

V poslovnih procesih nastopajo tudi različni udeleženci. S to besedo navadno mislimo na človeške vire, lahko pa jo uporabimo tudi za opis potrebnih računalniških virov.

Aktivnosti v procesu se lahko izvajajo zaporedno, vzporedno ali pa kot kombinacija obeh. Pri izvajanju pa se lahko uporabijo tudi zanke, ki omogočajo večkratno izvajanje ene ali več aktivnosti, dokler ni izpolnjen nek določen kriterij.



Slika 1: Primer procesa z več aktivnostmi [1]

3.1.2 Upravljanje s poslovnimi procesi

Upravljanje s poslovnimi procesi predstavlja v slovenščino preveden angleški izraz »Business Process Management«, ali krajše BPM. To je disciplina, ki se ukvarja z modeliranjem, avtomatizacijo in optimizacijo poslovnih procesov s ciljem večje prilagodljivosti, povečanja storilnosti in učinkovitosti ter posledično večje dobičkonosnosti podjetja.

Nekoliko zmede prinaša še drug pomen kratice BPM. Ta se uporablja tudi za poimenovanje računalniških sistemov, ki ponujajo računalniško podporo izvajanju te discipline.

Na upravljanje poslovnih procesov lahko gledamo iz dveh zornih kotov:

- BPM v organizaciji združbe,
- BPM v računalniških sistemih.

3.1.2.1 BPM v organizaciji združbe

Vsaka organizacija uporablja in izvaja poslovne procese za svoje delovanje. Pri tem morda najprej pomislimo samo na poslovni proces izdelave nekega izdelka ali izvajanja storitve. Vendar so dejansko prisotni še drugi poslovni procesi, vse od zaposlitve in uvajanja novega zaposlenca, pa ne nazadnje do naročila novega svinčnika.

Tako se upravljanje s poslovnimi procesi v organizaciji ukvarja z neprestanim postopki spreminjanja in prilagajanja poslovnih procesov, uvajanjem novih ter avtomatizacijo in optimizacijo obstoječih procesov. Pri teh postopkih se je dobro osredotočiti na znanje, ki ga imajo zaposlenci v tej združbi. Le ti najbolje poznajo obstoječo organizacijo ter obstoječe poslovne procese in so tako najbolj kompetentni za iskanje možnosti za izboljševanje. Veliko teh izboljšav lahko naredimo brez poseganja v obstoječe informacijske tehnologije.

Cilj dobrega upravljanja s procesi je učinkovito in produktivno organizirana združba.

3.1.2.2 BPM v računalniških sistemih

Pri BPM v računalniških sistemih se ukvarjamo s programsko opremo, ki omogoča računalniško podporo za upravljanje poslovnih procesov. Takšni sistemi podpirajo vse faze življenjskega cikla BPM. Le te so opisane v naslednjih poglavjih.

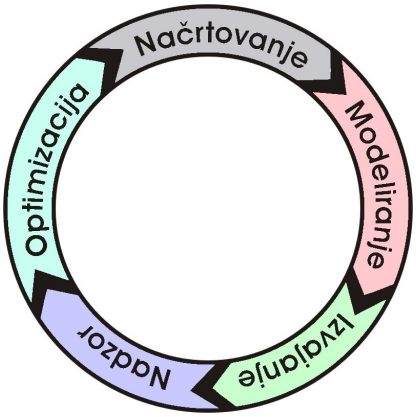
Uvajanje BPM v računalniške sisteme govori o dveh ločenih delih, in sicer:

- o načinu prenašanja podatkov med različnimi deli informacijskega sistema znotraj podjetja, kot tudi o povezovanju z drugimi podjetji. Ti prenosi naj bi bili elektronski in seveda samodejni. V tem primeru se pogosto omenja termin storitveno usmerjene arhitekture SOA (ang. Service Oriented Architecture), ta pa je tesno povezana s spletnimi storitvami. Ker pa skoraj v vsaki organizaciji srečamo tudi zastarele sisteme, moramo omogočiti tudi druge načine povezovanja.
- o načrtovanju in modeliranju poslovnih procesov kot osnovi za njihovo izboljševanje.

In ravno v teh dveh stopnjah bomo uporabljali delovne tokove (ang. Workflow), katerim je posvečen večji del te naloge.

Da so računalniški sistemi za podporo BPM pomembna tržna niša, dokazuje raziskava podjetja IDC, ki napoveduje več kot 10 odstotno letno rast prodaje takšnih sistemov.

3.1.3 Življenjski cikel BPM



Slika 2: Življenjski cikel BPM [2]

Življenjski cikel BPM sestoji iz petih stopenj in sicer:

- Načrtovanje
- Modeliranje
- Izvajanje
- Nadzor in spremljanje
- Optimizacija

3.1.3.1 Načrtovanje

V fazi načrtovanja se odločamo, katere poslovne procese bomo vključiti v BPM. Le te želimo avtomatizirati, prilagoditi ali pa le natančno opisati njihov potek, sodelujoče akterje in sodelovanje z drugimi procesi.

V stopnji načrtovanja preiskujemo in opisujemo obstoječe procese. Pri tem smo še posebej pozorni na:

- potek procesa,
- sodelujoče akterje,
- potek alarmiranja in obveščanja,
- standardizacijo postopkov,
- postopke prehajanja odgovornosti med akterji.

3.1.3.2 Modeliranje

V stopnji modeliranja uporabimo teoretični model, v katerega vpeljemo konkretne spremenljivke. S tem mislim na primer na potrebne vire, določene cene itd.

Nato izvajamo »kaj če« analizo, kjer se sprašujemo, kako vpliva sprememba katere izmed spremenljivk na naš poslovni proces. Takšno teoretično modeliranje nam lahko prihrani veliko stroškov, ki bi jih imeli z večkratnim popravljanjem naših že izdelanih procesov.

3.1.3.3 Izvajanje

Najpreprostejši način za avtomatizacijo našega poslovnega procesa je najem ali nakup programske opreme, ki omogoča izvajanje zahtevanih korakov. Žal pa je navadno realnost veliko bolj kompleksna. Tako te aplikacije le redko omogočajo natančno takšne postopke izvajanja, kot smo si jih zamislili. Tako se navadno odločamo za kombinacije med aplikacijsko podporo in sodelovanjem ljudi.

Kot odgovor na to potrebo se v aplikacije dodaja podpora za uporabnikom razumljivo grafično opisovanje poslovnih procesov. Ta pa se nato lahko pretvori v obliko, ki nam omogoča izvajanje v naši aplikaciji. Na ta način lahko določene korake izvaja računalnik sam, pri preveč kompleksnih korakih pa zahteva sodelovanje uporabnika. Za takšen način dela potrebujemo kar se da fleksibilno programsko opremo.

3.1.3.4 Nadzor in spremljanje

Nadzorovanje in spremljanje poslovnega procesa nam omogoča vpogled v njegovo trenutno stanje. Na tej podlagi izdelujemo statistike in spremljamo napredovanje. Na primeru sklepanja leasing poslov bi lahko spremljali, koliko informativnih ponudb je bilo izdanih strankam, kot tudi ali se je določena ponudba že prebila skozi postopek odobritve in je pripravljena za izvedbo posla.

Poseben izraz v nadzorovanju je **procesno rudarjenje** (ang. Process mining). Cilj rudarjenja je analiza dnevnikov izvajanja delovnih procesov. Rezultate analize primerjamo z izdelanim procesnim modelom. To nam omogoča odkrivanje nedoslednosti med izdelanimi delovnimi procesi in izdelanim modelom. Olajša pa nam tudi iskanje ozkih grl poslovnega procesa.

3.1.3.5 Optimizacija

Zelo pomembna je tudi zadnja stopnja - optimizacija, ki predstavlja nekakšen logičen zaključek vseh predhodnih štirih. Tako uporabimo vse podatke o učinkovitosti delovanja modela iz faze modeliranja, kot tudi faze nadziranja. Rezultat pa so raziskana ozka grla našega poslovnega procesa in možnosti za potencialne izboljšave.

3.1.4 Uvedba BPM v praksi

Koraki v življenjskem ciklu BPM se lahko izvajajo iterativno. Število iteracij pa je odvisno od finančnih in ekonomskih omejitev. Zato je zelo pomembno, da še pred pričetkom uvajanja BPM-a prepoznamo področja, ki nam bodo omogočila največjo stopnjo napredka.

Preden pa se lotimo računalniške avtomatizacije poslovnih procesov, morajo biti le ti kar se da učinkoviti. Za izboljšave in prilagoditve pa morajo v prvi vrsti poskrbeti zaposlenci v združbi, kamor BPM uvajamo. Le ti so namreč dovolj kompetentni in odgovorni za oblikovanje poslovnih procesov in definiranje poslovnih pravil.

Da je učinkovitost poslovnih procesov zelo pomembna, še preden se lotimo avtomatizacije in uvajanja BPM v združbo, je poudaril tudi soustanovitelj korporacije Microsoft, Bill Gates, ko je zapisal:

> *»Prvo pravilo vsake tehnologije je, da bo vsaka avtomatizacija učinkovite operacije povečala učinkovitost. Drugi pravilo pa pravi, da bo avtomatizacija, uporabljena na neučinkovitih operacijah, povečala neučinkovitost.«[3]*

Ravno v velikih poslovnih sistemih se osnovni procesi poslovanja niso spreminjali in prilagajali že vrsto let. Nekateri pa so ostali nespremenjeni že več generacij. Primer branž, kjer se poslovni procesi le redko spreminjajo, so ravno bančništvo, zavarovalništvo in finance.

3.1.5 Kdaj uporabiti BPM

Na vprašanje, kdaj je uporaba BPM smiselna, je težko odgovoriti na povsem splošen način. Le to je odvisno od organizacije do organizacije, od situacije do situacije. Vendar pa obstaja seznam lastnosti in pogojev, ki kažejo na smiselnost njegove uporabe. Med njimi so najpomembnejši naslednji:

- velika količina podobnih in ponavljajočih se poslovnih procesov;
- velika količina zelo jasnih poslovnih procesov, ki potekajo preko več udeležencev, kjer vsak izmed njih doda nekaj dodane vrednosti;

- potreba po spremljanju poslovnega procesa v realnem času, tako da vedno vemo, kateri procesi se izvajajo in v kakšnem stanju so;
- potreba po hitrem izvajanju poslovnega procesa;
- potreba po izvajanju večje količine kalkulacij v postopku izvajanja procesa;
- težave zaradi hitre rasti obsega poslovanja in hitro menjavanje zaposlenih;
- potreba po veliki prilagodljivosti, ki omogoča organizaciji prilagajanje na poslovne priložnosti;

3.1.6 Ključni dejavniki za uspeh projektov BPM

Uvedba v BPM se pogosto izkaže kot veliko bolj kompleksen problem, kot pa je to videti na prvi pogled. Pogosto vsebuje spremembe odnosov med ključnimi udeleženci, tako znotraj organizacije kot tudi zunaj nje. Na ta način je vsak projekt unikaten, vendar lahko vseeno opredelimo nekaj ključnih dejavnikov, potrebnih za uspešno uvedbo BPM:

- ***Močna podpora vodstva***. Brez jasne podpore vodstva podjetja se uvedbe BPM v organizacijo ne moremo lotiti;
- ***Izkušnje projektnih vodij z BPM***. Le te so zelo pomembne za uspešen zaključek projekta;
- ***Navezava s strategijo organizacije***. Vsak projekt mora doprinesti neko dodano vrednost k izvajanju strategije organizacije. Če kakšen projekt ne izpolnjuje tega pogoja, ga rajši niti ne pričnemo;
- ***Strinjanje ostalih udeležencev***. Vse procese izvajajo ljudje ali pa računalniki s podporo ljudi. Tako bodo prav ti udeleženci odločilno pripomogli k uspehu ali polomu projekta. Vsako nestrinjanje udeležencev precej poviša možnosti za neuspeh.

3.1.7 Orodja za upravljanje s poslovnimi procesi

Pri uvajanju BPM-a je smiselno uporabiti podporno programsko opremo. Trenutno je na tržišču veliko aplikacij, ki omogočajo izvajanje različnih delov BPM, kot so modeliranje in simulacija poslovnih procesov, njihovo vrednotenje, iskanje ozkih grl itd. Ta orodja navadno omogočajo grafično predstavitev modelov poslovnih procesov. Prav tako pa omogočajo spremljanje izvajanja, beleženje časa izvajanja posameznih aktivnosti, stroške in potrebne vire. Pri tem pa lahko poizkušamo ugotoviti paralelizme in omogočiti hkratno izvajanje poslovnih procesov na več mestih.

3.2 Windows Workflow foundation

3.2.1 Delovni tokovi

Delovni tokovi definirajo postopke, odločitve ter zbirke pravil in so pripravljeni na komunikacijo z zunanjimi sistemi. Osnovno enoto predstavlja aktivnost. Skupek le teh pa opisuje problem iz realnega sveta.

Termin podatkovni tok predstavlja definiranje postopkov in poslovnih pravil. Pri tem uporabljamo aktivnosti, ki jih med seboj povezujemo v smiselno zaporedje.

Problemi, rešeni na takšen način, so veliko bolj pregledni, rešitve pa lažje tudi spreminjamo, nadgrajujemo in optimiziramo.

Delovni tokovi so navadno predstavljeni v grafični obliki, kar nam omogoča neposredno vključevanje poznavalcev specifične problemske domene v proces razvoja in vzdrževanja programske opreme, pa čeprav le-ti niso ravno razvijalci.

Za definiranje in izvajanje delovnih tokov uporabljamo sisteme za upravljanje delovnih tokov (ang. Workflow Management Systems ali WMS). Eden izmed njih je tudi Windows Workflow foundation, ki je bolj natančno opisan v naslednjem poglavju.

Aktivnosti so lahko avtomatizirane oz. samodejne, ali pa ročne. Samodejne izvaja WMS samostojno. Ročne pa predstavljajo delo, ki ga WMS ne zna izvajati sam in za to potrebuje zunanje udeležence. To je lahko neka zunanja aplikacija ali pa kar človek.

Izvajanje delovnega toka poteka tako, da WMS najprej ustvari primerek (ang. instance) delovnega toka, ki ga zgradi po predhodni definiciji le tega. Nato po vrstnem redu, določenem z definicijo delovnega toka, izvaja posamezne aktivnosti.

Istočasno lahko WMS izvaja več različnih primerkov istega delovnega toka, ki pa se zaradi različnih vhodnih parametrov različno obnašajo.

3.2.2 Kaj je Windows Workflow foundation

Windows Workflow foundation je programski model, ki razvijalcem ponuja že izdelano infrastrukturo za uporabo delovnih tokov znotraj aplikacije, ki jo razvijajo. Uradna kratica za Windows Workflow foundation je WF, kljub temu pa v nekateri literaturi zasledimo tudi kratico WWF. Uporaba slednje ni pravilna, saj se uporablja za povsem druge namene, kot recimo okrajšava za organizacijo »World Wildlife Fund«.

Ta infrastruktura vsebuje serijo že razvitih aktivnosti, podporo več tipom delovnih tokov, postopke za začasno shranjevanje toka v podatkovno bazo in kasnejše ponovno »prebujanje«. Razvijalci lahko uporabijo še možnost sledenja poteku tokov, uporabi transakcij in določitvi odzivov na morebitne napake v delovanju ter uporabe transakcij.

S pomočjo tehnologij ASP.NET oz. Windows Communication Foundation pa lahko omogočimo uporabo že razvitih delovnih tokov tudi v drugih sistemih preko standardiziranega vmesnika, kot so spletne storitve.

3.2.3 Vrste delovnih tokov

Poznamo dve vrsti delovnih tokov, in sicer ***sekvenčne delovne tokove*** in ***delovne tokove tipa končni avtomat***. Pravilna izbira je zelo pomembna, vendar ni vedno povsem enostavna. Prav tako je zelo težaven prehod iz rešitve, ki uporablja sekvenčne delovne tokove na rešitev s končnim avtomatom in obratno. Zato je poznavanje prednosti in slabosti ene in druge vrste delovnih tokov še bolj pomembno.

3.2.3.1 Sekvenčni delovni tokovi

Sekvenčni delovni tokovi omogočajo razvoj fiksnega zaporedja aktivnosti. Tako ima vsak tok jasno definiran začetek, postopek in zaključek. Vsebuje lahko razmejitve in zanke, prav tako se lahko odziva na dogodke. Po končanem izvajanju lahko vrne rezultat. Za razvoj lahko uporabimo nabor že izdelanih aktivnosti, ali pa uporabimo lastne.

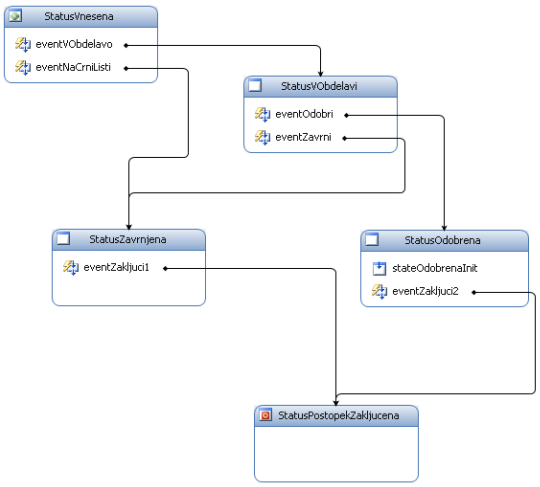
3.2.3.2 Delovni tokovi tipa končni avtomat

Za razliko od sekvenčnega, kjer gre za fiksno zaporedje aktivnosti, pa pri delovnih tokovih tipa končni avtomat govorimo s stanjih. Vsako stanje toka lahko odraža dejansko stanje v našem delovnem procesu. Tako lahko imamo stanja ponudbe, kot so »vnesena«, »v obdelavi«, »odobrena«, »zavrnjena«, itd.

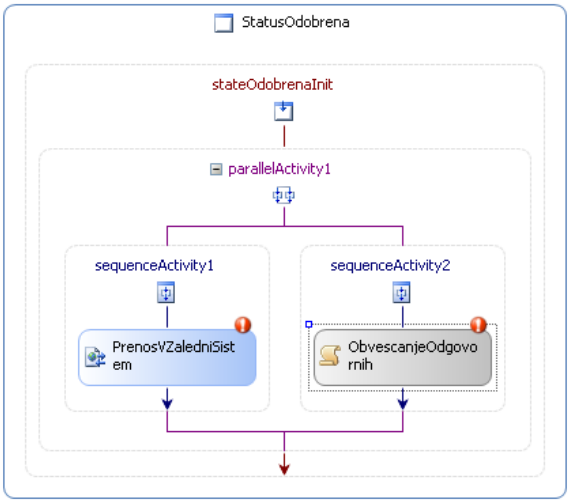
Za vsako stanje določimo dogodke, povezane s tem stanjem in jim določimo prehode v nova stanja. Če uporabim prejšnji primer, lahko določimo prehode iz »vnesena« v stanje »v obdelavi«, ali pa (recimo da gre za stranko, ki je neredni plačnik pri drugih pogodbah) kar direktno v stanje »zavrnjena«. Če pa je pogodba v »obdelavi«, jo lahko prestavimo v »odobrena« ali »zavrnjena«.

V vsakem stanju lahko določimo več različnih sekvenčnih delovnih tokov, in sicer:

- delovni tok, ki se izvede ob prehodu v določeno stanje,
- delovni tok, ki se izvede ob zapustitvi tega stanja,
- delovni tok, ki se izvede glede na trenutno stanje in glede na trenutni dogodek.



Slika 3: Primer delovnega toka tipa končni avtomat

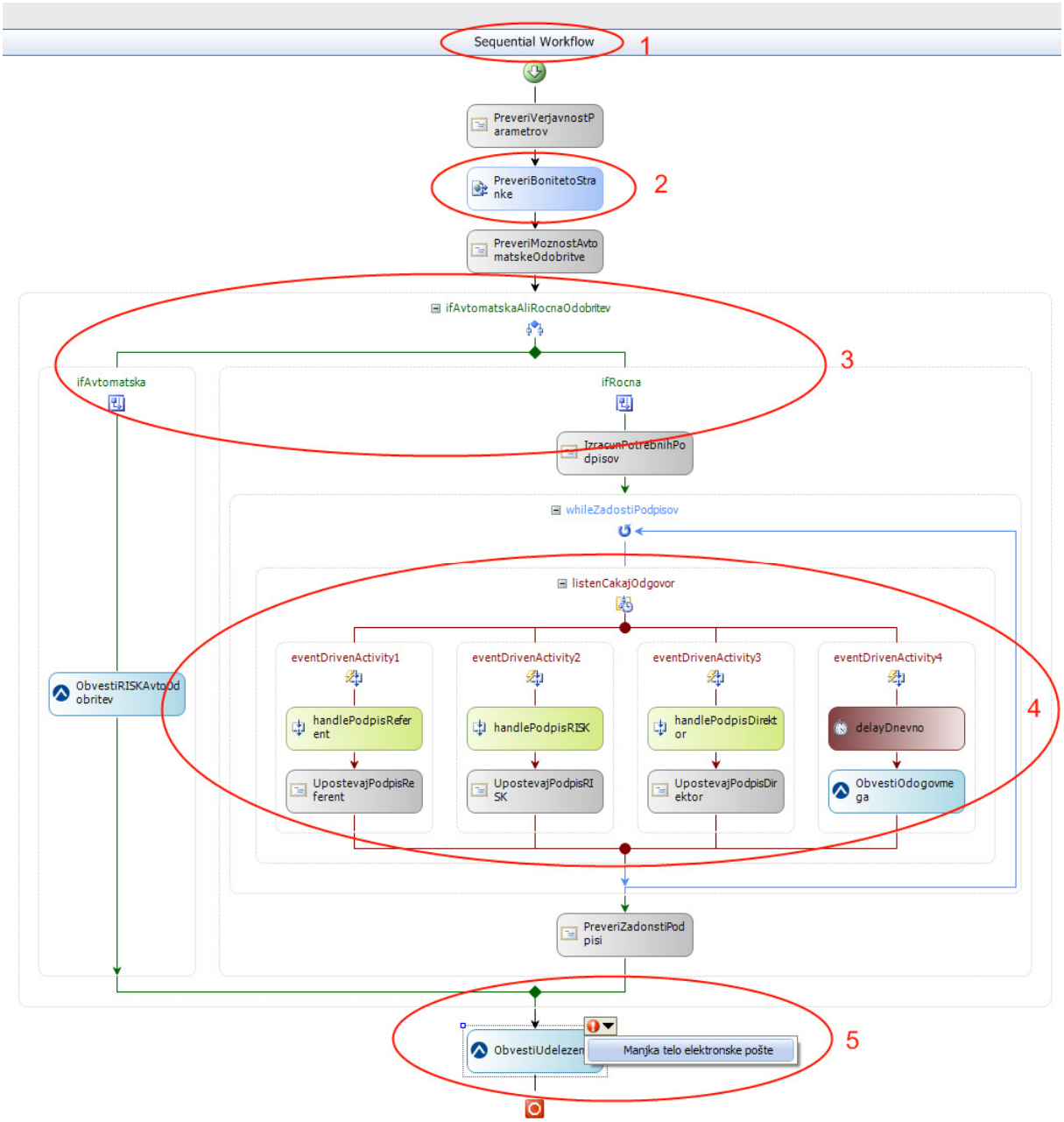


Slika 4: Primer odziva delovnega toka na prehod v status Odobrena

3.2.4 Prikaz urejevalnika delovnih tokov

Del ogrodja je tudi urejevalnik za delovne tokove. Le tega uporabljamo neposredno v orodju Microsoft Visual Studio, ki nam omogoča preprostejši način .NET aplikacij. Z nekaj truda pa lahko povsem enak urejevalnik gostimo tudi v naši lastni aplikaciji. To je še posebej priročno, kadar želimo omogočiti urejanje tudi uporabnikom naših rešitev, ki pa praviloma nimajo nameščenega prej omenjenega orodja.

Delovni tok v urejevalniku vidimo na spodnji sliki.



Slika 5: Primer delovnega toka

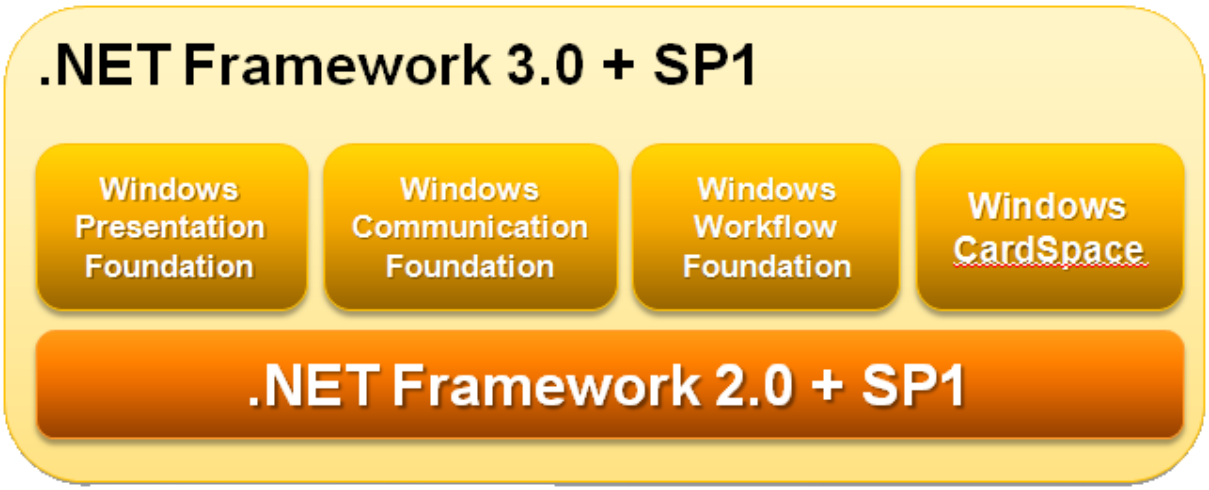
V gornjem primeru sem označil tudi nekaj pomembnejših delov, ki jih še dodatno pojasnjujem, kot sledi:

1. V glavi urejevalnika vidimo tip delovnega toka. V primeru sem uporabil sekvenčni delovni tok. Poznamo pa tudi tako imenovani delovni tok tipa končni avtomat.
2. Pravokotniki, kot je npr. označen s številko 2, predstavljajo osnove aktivnosti (ang. activity). Natančneje gre za InvokeWebServiceActivity, ki nam omogoča klice spletnih storitev.

3. Poleg osnovnih aktivnosti poznamo tudi sestavljene aktivnosti (ang. composite activity), ki lahko vsebujejo tudi druge osnovne ali sestavljene aktivnosti. V tem primeru imamo vejitev, ki se na podlagi rezultata poslovnega pravila odloča, ali se bo izvedla samodejna odobritev, ali pa bo morala ponudba potovati skozi celoten postopek ročnega podpisovanja.
4. Še ena sestavljena aktivnost, ki povzroči, da delovni tok prične čakati na vhod iz gostujoče aplikacije. V tem primeru čakamo, da bo uporabnik ponudbo odobril oz. podpisal ali pa jo zavrnil. V zadnji veji pa sem dodal tudi časovnik, ki omogoča nadaljevanje toka tudi če gostujoča aplikacija ne posreduje podatkov. V prikazanem primeru bomo, če en dan ne bo nobene odobritve ne zavrnitve, uporabnikom poslali opomnik preko elektronske pošte.
5. Vsaka aktivnost ima določen seznam obveznih in neobveznih parametrov. V primeru, da kateri izmed obveznih parametrov ni izpolnjen (pri tem lahko uporabimo tako konstanto, kot sklic na globalno definirano spremenljivko, kot tudi rezultat izvajanja katerega izmed predhodnih aktivnosti), nas urejevalnik na to opozori z rdečim klicajem. Če premaknemo kazalec miške nad njega, pa nam urejevalnik prikaže še dodatne informacije o manjkajočih parametrih. Dokler ne odpravimo vseh klicajev, prevod in uporaba delovnega toka ni mogoča.

3.2.5 .NET Framework 3.0

Programski model **Windows Workflow Foundation** se prvič pojavi v ogrodju .NET Framework 3.0, znanem tudi kot **WinFX**. Ta med drugim omogoča definiranje, upravljanje in izvajanje delovnih tokov. Na voljo je kot ločena namestitev za operacijske sisteme Windows XP in 2003 Server. V operacijski sistem Windows Vista in Windows Server 2008 pa je že vključen.



Slika 6: .NET Framework 3.0 sklad

V okviru odprto kodnega projekta Mono pa lahko večji del ogrodja .NET Framework 3.0 uporabljamo tudi na drugih operacijskih sistemih, kot je npr. Linux. Po podatkih iz domače strani projekta je na področju delovnih tokov implementiranih 70% aktivnosti [4]. Žal pa so ostali pomembnejši deli, kot sta podpora zunanji komunikaciji in pravilom le delno podprti. Trenutno pa ni podpore med drugim za sledljivost in vzdrževanje stanj.

Zadnja verzija ogrodja .NET Framework 3.5 WF ne prinaša nič posebno novega. Najpomembnejša novost je izboljšana integracija z Windows Communication Foundation. V ta namen so dodali nekaj novih aktivnosti.

3.2.6 Windows Communication Foundation

Windows Communication Foundation (WCF), znan tudi kot **Indigo**, je izšel novembra 2005 kot del ogrodja .NET Framework 3.0. Njegov namen je združiti različne načine med aplikacijske komunikacije v en standardiziran, storitveno usmerjen model. Tako imamo na voljo komunikacijo na podlagi SOAP protokola (spletne storitve) in XML zapisa, ki nam zagotavlja največjo možno stopnjo standardizacije in možnost povezovanja različnih sistemov. Pomembno je tudi to, da WCF poleg osnovnega Web Services 1.0 standarda, v celoti podpira tudi tako imenovane WS-* standarde. Kot alternativo lahko izberemo tudi optimizirano binarno komunikacijo, omejeno le na Windows okolja. Za asinhrono komunikacijo imamo na voljo tudi podporo za različne sporočilne vrste.

Z novo verzijo, ki je del ogrodja .NET Framework 3.5 pa je SOAP protokolu dodan še JSON. To nam omogoča uporabo storitev, razvitih s pomočjo WCF tudi v AJAX spletnih

aplikacijah. Prav tako pa je z novo verzijo na voljo veliko boljša integracija WCF in WF, kar nam bo zelo koristilo pri implementaciji WF v obstoječe aplikacije.

Vsaka WCF storitev sestoji iz treh delov. Najprej je ***storitveni razred***, kjer razvijemo vso logiko, ki jo storitev izvaja. Definiramo tudi t.i. storitveno pogodbo (ang. Service Contract), ki določa operacije, ki jih zna storitev izvajati ter njihove vhodne in izhodne parametre. Za opisovanje tipov parametrov se uporablja podatkovna pogodba (ang. Data Contract). Če za tipe uporabljamo le osnovne podatkovne tipe, za njo poskrbi WCF sam. V primeru, da uporabljamo kompleksne tipe, pa jih moramo označiti z Data Contract atributom in WCF bo sam poskrbel za njihov zapis v standardizirano obliko.

Naslednji del je ***okolje***, ki gosti našo storitev. Tukaj bomo najpogosteje uporabili gostovanje v Windows storitvi (ang. Windows Service), ali pa bomo uporabili strežnik IIS. Windows storitev predstavlja dolgo izvajajoči proces v operacijskem sistemu Windows, ki ne vsebuje uporabniškega vmesnika. Strežnik IIS pa je spletni strežnik podjetja Microsoft.

Nazadnje moramo določiti še tako imenovano ***zaključno točko*** (ang. Endpoint). Ta predstavlja naslov, kjer bo naša storitev na voljo porabnikom, in protokol, ki ga uporabljamo za prenos sporočil. Najpogosteje se uporabljajo protokoli SOAP preko HTTP, SOAP preko TCP in SOAP preko sporočilnih vrst.

Slika 7: Arhitektura WCF. Vir: Andru's WebLog [5]

3.2.7 Podobnosti WF z BizTalk Server

BizTalk server je rešitev za tako imenovano business to business povezovanje različnih sistemov podjetja Microsoft. V nekaterih virih [6] govorijo kar o BPM strežniku.

Značilnosti BizTalk serverja so naslednje:

- Za povezovanje aplikacij uporablja tako imenovane priključke. Na voljo so priključki za povezovanje neposredno na aplikacije (npr. Siebel, SAP, …), podatkovne baze (MS SQL Server, Oracle, DB2) in ostale tehnologije (J2EE);
- Možnost pisanja poslovnih pravil v psevdo angleškem jeziku;
- Spremljanje poslovnih pravil (angleško Business Actvity Monitoring – BAM), ki omogoča grafičen pregled nad trenutnim izvajanjem procesov;
- S pomočjo drugih Microsoft produktov, kot je pisarniška zbirka Office, program za urejanje obrazcev Infopath in sistem za podporo skupinskemu delu Sharepoint, omogoča interakcijo neposredno z zunanjimi uporabniki;
- Podpora standardu za elektronsko izmenjavo podatkov EDI (Electronic Data Interchange);
- Možnost grafičnega modeliranja poslovnih procesov z uporabo razvijalskega orodja Visual Studio;

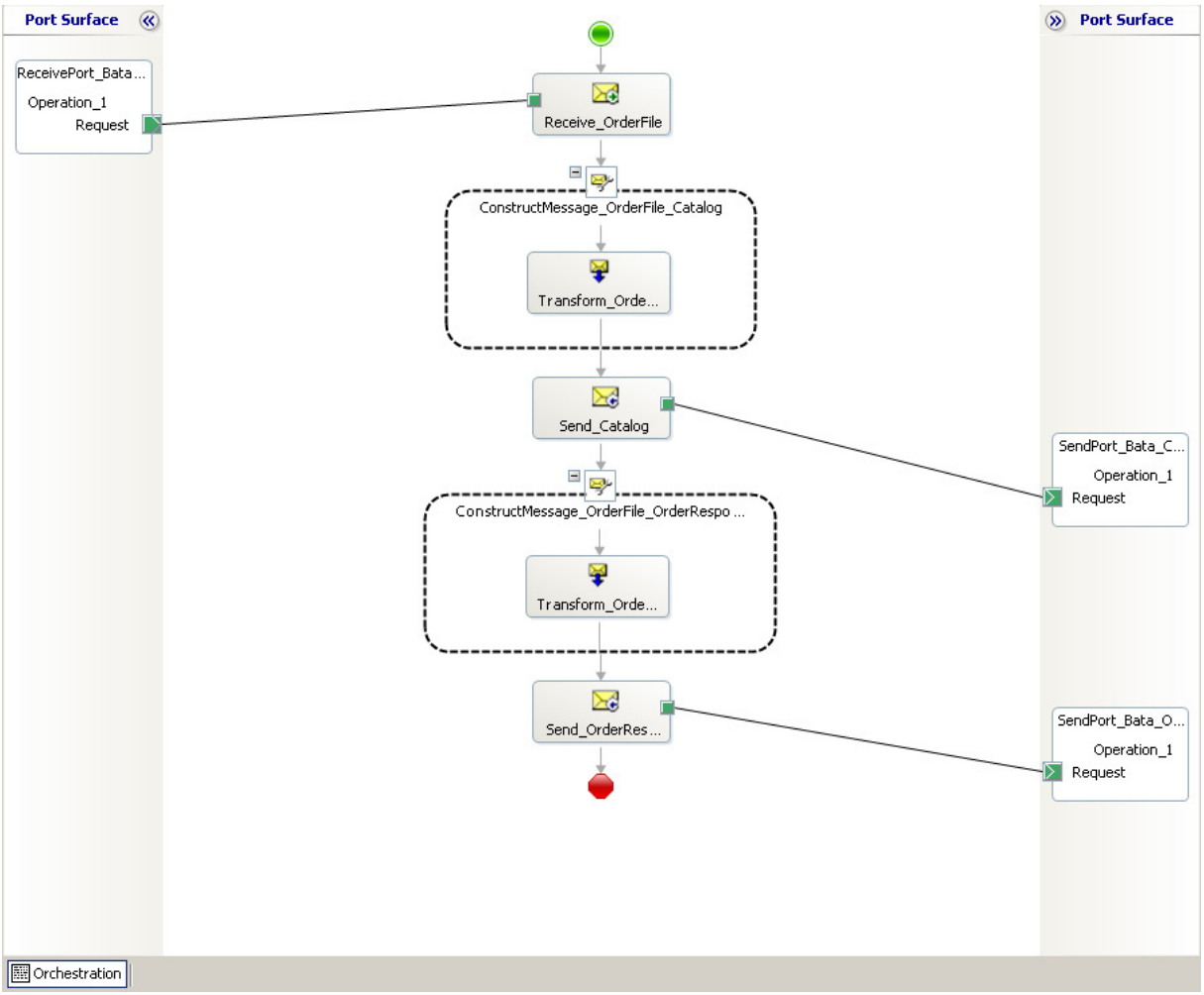
Če primerjamo WF z rešitvijo BizTalk, ugotovimo nekaj podobnosti, vendar tudi veliko razlik. Kot sem že omenil, se lahko s pomočjo priključkov BizTalk povezujemo na zelo širok krog zunanjih aplikacij. WF je v tem pogledu skromnejši. V paketu nam ponuja le možnost povezovanja preko spletnih storitev, v različici 3.5 pa še preko WCF. Ostale povezave pa bomo morali razviti sami.

Tako BizTalk kot tudi WF že vsebujeta podporo za sledenje izvajanja delovnih tokov. Pa vendar je BizTalk tudi v tem pogledu veliko bolj napreden. Prav tako vsebuje že orodja za nadzor že med izvajanjem.

Načrtovanje delovnih tokov je v obeh tehnologijah precej podobno. V obeh primeru na strukturiran način dodajamo aktivnosti in jih uvrščamo na določeno mesto v strukturi. To mesto pa določa, kdaj se bo aktivnost izvedla, katere aktivnosti se bodo izvedle pred njo, katere paralelno z njo in katere za njo. Obe tehnologiji za modeliranje uporabljata razvojno okolje Microsoft Visual Studio. Le da ogrodje WF omogoča gostovanje urejevalnika tudi v lastnih aplikacijah.

Zelo pomembna prednost WF pred BizTalk Server-jem pa je možnost, da le tega vgradimo neposredno v svojo aplikacijo, ki za svoje delovanje potrebuje delovne tokove. Najprej nam to olajša in pospeši razvoj same aplikacije, še pomembnejša prednost pa je olajšana distribucija aplikacije. Vse kar potrebujemo za zagon delovnega toka, razvitega v WF, je namreč vključeno v .NET Framework, zato namestitev dodatne programske opreme ni potrebna. Delovni tokovi, razviti v BizTalk-u pa za svojo izvajanje potrebujejo BizTalk strežnik. Tako moramo pred izvajanjem teh delovnih tokov poskrbeti za pravilno namestitev, nastavitve omenjenega produkta in nakup licenc.

Za razliko od BizTalk Server-ja, ki je v celoti komercialni produkt, za katerega potrebujemo dodatne licence, je .NET Framework že del operacijskega sistema Windows Vista in Windows 2008 Server oz. je na voljo kot brezplačen dodatek za operacijske sisteme Windows 2000 in novejše.



Slika 8: Primer delovnega toka v BizTalk Server 2006 R2

3.2.8 Možnosti uporabe WF in povezovanje z drugimi sistemi

Kot sem omenil že v prejšnjih poglavjih je WF samo programski model, ki nam omogoča razvoj in izvajanje delovnih tokov. Tako moramo sami poskrbeti za gostitelja WF izvajalnega okolja. To nam olajša distribucijo naše rešitve, ki uporablja WF. Tako je edini pogoj nameščen .NET Framework 3.0 ali novejši.

Tak pristop nam omogoča več načinov izvajanja podatkovnih tokov. Vsak izmed njih ima določene prednosti glede na naše potrebe.

3.2.8.1 Izvajanje znotraj aplikacije

Takšnega načina se bomo pogosto posluževali, ko bomo razvijali lastno rešitev, napisano za .NET Framework. Za zagon izvajanja toka potrebujemo le nekaj vrstic programske kode. Nimamo težav s serializacijo objektov, do katerih pride, če želimo objekte spremeniti v obliko, primerno za pošiljanje po omrežju. Zaradi vsega tega pa je boljša tudi odzivnost celotnega sistema.

Zelo pomembna prednost pa je tudi možnost dvosmerne komunikacije. Tako lahko s pomočjo posebnih aktivnosti in definicij razredov omogočimo, da podatkovni tok sam zahteva dodatne podatke od svojega gostitelja, ali pa ga recimo obvešča o napredku izvajanja.

V kolikor potrebujemo storitve delovnega toka samo znotraj ene aplikacije in se le te ne spreminja pogosto, je takšen način uporabe WF pravilen.

Zaradi specifike delovanja spletnih aplikacij ima gostovanje WF v ASP.NET aplikaciji nekatere omejitve. Več o tem v poglavju 4.2.

3.2.8.2 Objava kot spletna storitev

Če naše podatkovne tokove objavimo kot spletne storitve, omogočimo njihovo uporabo preko standardiziranih vmesnikov. Na tak način lahko omogočimo uporabo logike najrazličnejšim aplikacijam, razvitih v poljubnih programskih jezikih, ki tečejo na različnih operacijskih sistemih. To nam omogoča večjo stopnjo ponovne uporabljivost izdelane poslovne logike. Tako se približamo značilnostim storitveno usmerjene arhitekture (SOA).

Pri razvoju moramo paziti, da imajo vsi tipi parametrov možnost zapisa v XML. To je namreč pogoj za prenašanje po omrežju, standard XML pa je osnova za standardizirano povezovanje različnih sistemov.

Pri uporabi takšnega načina objave izgubimo možnost dvosmerne komunikacije. Tako uporaba deluje po postopku zahteva – odgovor. Je pa lahko izvedba delovnega toka povezana z večjim številom zahtev in pripadajočih odgovorov. Ne moremo pa doseči, da bi sam delovni tok od svojega gostitelja zahteval posredovanje (dodatne podatke, odločitev uporabnika, ipd.).

Zelo pomembna prednost objave delovnih tokov preko spletne storitve je tudi centraliziran nadzor nad nameščanjem in izvajanjem. To je še posebej pomembno pri poslovnih pravilih, ki se pogosto spreminjajo, ker nam takšna uporaba prihrani nadgradnje posameznih klientov. Prav tako pa zagotovimo, da vsi klienti delujejo z enakimi pravili.

3.2.8.3 Objava preko WCF

S pomočjo WCF lahko združimo pozitivne lastnosti izvajanja delovnih tokov znotraj aplikacije in objave kot spletno storitev. Ta tehnologija, znana tudi pod imenom **Indigo**, nam olajša postavitev komunikacijske infrastrukture za izdelavo povezanih sistemov.

Z uporabo WCF nam bo na voljo centralizirana namestitev, izvajanje, nadziranje in vzdrževanje delovnih tokov. Uporabili pa bomo lahko več različnih protokolov za povezovanje. Pri tem bomo lahko izbirali med:

- optimiziranimi binarnimi zapisi, združljivimi samo z Windows okolji in standardiziranimi XML zapisi, ki je zaradi svojih dodatnih informacij nekoliko daljši;
- protokoli, ki podpirajo eno ali dvosmerno komunikacijo;
- protokoli, ki skrbijo za potrjevanje prejema podatkov in za njihovo integriteto;
- prenosi podatkov v šifrirani obliki ali kot golo besedilo.

Te parametre določamo glede na sisteme, ki bodo uporabljali storitve delovnih tokov, kot tudi glede na količino in občutljivost podatkov, ki jih bomo prenašali.

Če želimo naše delovne tokov izpostaviti navzven in omogočiti njihovo uporabo v več aplikacijah, bo izbor objave preko WCF najprimernejša izbira. V veliko pomoč pa nam bo tudi izboljšana integracija med WF in WCF v novejših različicah ogrodij .NET Framework.

3.2.9 Beleženje dnevnika

V življenjskem ciklu upravljanja s poslovnimi procesi je tudi nadzor in spremljanje delovanja poslovnih procesov. Rezultat le tega pa je vhodni podatek za stopnjo optimizacije, ki predstavlja naš končni cilj.

Na srečo so se te pomembnosti zavedali tudi snovalci Windows Workflow Foundation-a in so možnost beleženja podatkov o izvajanju vključili v samo ogrodje. Deluje kot dodatna storitev, ki jo vključimo v izvajalno okolje. Le ta samodejno beleži podatke o poteku izvajanja delovnega toka. Dodatne informacije pa lahko dodajajo tudi aktivnosti.

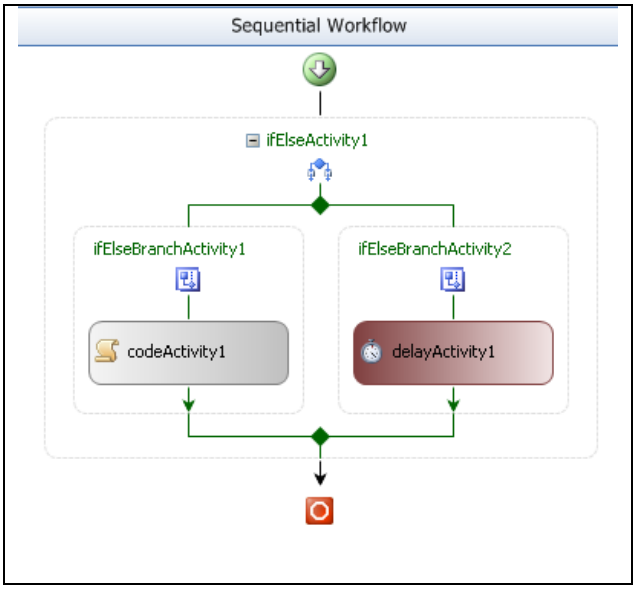
V ogrodju je na voljo storitev beleženja dnevnika v podatkovno bazo Microsoft SQL. Pripravljeno pa je tudi ogrodje za razvoj lastne storitve za beleženje dnevnika, kjer lahko sami izbiramo obliko zapisa in medij za hranjenje.

3.2.10 Zapis WF v obliki kode proti zapisu XAML

Windows Workflow foundation omogoča zapis delovnega toka na dva načina. V obliki programske kode, v kateri določamo, katere aktivnosti bomo naredili in kako so med seboj hierarhično povezane. Programsko kodo nato prevedemo v zbir (angleško assembly).

Drugi pristop pa je zapis delovnega toka v XAML obliki. To je deklarativni jezik za strukturiran opis vrednosti in objektov. Za zapis uporablja razširljivo obliko XML. Uporablja se tudi v drugih Microsoftovih tehnologijah od ogrodja .NET Framework 3.0 naprej. Primer je Windows Presentation Foundation, kjer se XAML uporablja za zapis elementov uporabniškega vmesnika, povezovanja le tega z izvori podatkov in dogodki.

Delovni tok, zapisan v XAML obliki lahko prevedemo v zbir, kot smo to naredili v primeru zapisa s programsko kodo. Lahko pa ga tudi pustimo v XAML obliki izven aplikacije in na takšen način omogočimo prilagajanje brez potrebe po ponovnem prevajanju celotne aplikacije.



Slika 9: Primer preprostega delovnega toka

```
this.CanModifyActivities = true;
System.Workflow.Activities.CodeCondition codecondition1 = new System.Workflow.Activities.CodeCon
this.delayActivity1 = new System.Workflow.Activities.DelayActivity();
this.codeActivity1 = new System.Workflow.Activities.CodeActivity();
this.ifElseBranchActivity2 = new System.Workflow.Activities.IfElseBranchActivity();
this.ifElseBranchActivity1 = new System.Workflow.Activities.IfElseBranchActivity();
this.ifElseActivity1 = new System.Workflow.Activities.IfElseActivity();

this.delayActivity1.Name = "delayActivity1";
this.delayActivity1.TimeoutDuration = System.TimeSpan.Parse("00:00:10");

this.codeActivity1.Name = "codeActivity1";
this.codeActivity1.ExecuteCode += new System.EventHandler(this.Code1);

this.ifElseBranchActivity2.Activities.Add(this.delayActivity1);
this.ifElseBranchActivity2.Name = "ifElseBranchActivity2";

this.ifElseBranchActivity1.Activities.Add(this.codeActivity1);
codecondition1.Condition += new System.EventHandler<System.Workflow.Activities.ConditionalEventA
this.ifElseBranchActivity1.Condition = codecondition1;
this.ifElseBranchActivity1.Name = "ifElseBranchActivity1";

this.ifElseActivity1.Activities.Add(this.ifElseBranchActivity1);
this.ifElseActivity1.Activities.Add(this.ifElseBranchActivity2);
this.ifElseActivity1.Name = "ifElseActivity1";

this.Activities.Add(this.ifElseActivity1);
this.Name = "WF_Koda";
this.CanModifyActivities = false;
```

Slika 10: Zapis preprostega delovnega toka v obliki kode

```
<SequentialWorkflowActivity x:Class="WorkflowProject1.WF_XAML" x:Name="WF_XAML" xmlns:x="http://schemas.micr
  <IfElseActivity x:Name="ifElseActivity1">
    <IfElseBranchActivity x:Name="ifElseBranchActivity1">
      <IfElseBranchActivity.Condition>
        <CodeCondition Condition="Test1" />
      </IfElseBranchActivity.Condition>
      <CodeActivity x:Name="codeActivity1" ExecuteCode="Code1" />
    </IfElseBranchActivity>
    <IfElseBranchActivity x:Name="ifElseBranchActivity2">
      <DelayActivity TimeoutDuration="00:00:10" x:Name="delayActivity1" />
    </IfElseBranchActivity>
  </IfElseActivity>
</SequentialWorkflowActivity>
```

Slika 11: Zapis delovnega toka v obliki XAML

3.3 Opis ostalih tehnologij, omenjenih v nalogi

3.3.1 Microsoft Internet Information Services (IIS)

Microsoft Internet Information Services, ali krajše IIS, predstavlja spletni strežnik proizvajalca programske opreme Microsoft. Sestavljen je iz več Windows storitev (ang. Windows Services) in deluje na operacijskem sistemu Windows. Vključuje strežnike za več protokolov, kot so HTTP, HTTPS, FTP, SMTP, itd.

IIS je bil prvič predstavljen kot dodatek za Windows NT 3.51. Trenutno pa je na voljo različica 7.0, ki je del operacijskega sistema Windows Server 2008 in Windows Vista.

Po podatkih podjetja za internetne raziskave Netcraft[7], naj bi bil IIS na drugem mestu po popularnosti spletnih strežnikov. Po podatkih iz junija 2008 le ta skrbi za gostovanje dobrih 35% vseh spletnih mest.

3.3.2 ASP.NET

ASP.NET je ogrodje za razvoj spletnih aplikacij, ki ga razvija Microsoft. Omogoča tudi gradnjo spletnih storitev. Prvič je bil predstavljen, kot del .NET Framework-a 1.0 v začetku leta 2002, kot izboljšava tehnologije ASP.

ASP.NET aplikacije lahko razvijamo v poljubnem programskem jeziku, ki podpira .NET Framework. Aplikacije, razvite v ASP.NET za izvajanje potrebujejo spletni strežnik, ki podpira izvajanje ASP.NET rešitev. V večini primerov je to Microsoft IIS 5.0 ali novejši. Zelo aktiven pa je tudi projekt Mono, cilj katerega je med drugim tudi podpora izvajanju ASP.NET rešitev na drugih spletnih strežnikih.

3.3.3 Spletne storitve

Spletne storitve (ang. Web Service) so po definiciji konzorcija W3C »programski sistemi, ki so narejeni tako, da omogočajo sistemsko neodvisno interakcijo preko omrežja«. Spletne storitve tako predstavljajo standardizirani spletni vmesnik, ki omogoča klice oddaljenih metod na oddaljenem sistemu. Poglavitna prednost takšnega klica pred ostalimi načini je ravno standardizacija W3C, ki predstavlja vodilno mednarodno organizacijo za postavitve standardov na svetovnem spletu. Tako sodelovanje ni več pogojeno z uporabo istega programskega jezika, spletnega strežnika ali operacijskega sistema. Odjemalci s strežnikom navadno komunicirajo v obliki XML sporočil. Za opis storitve se uporablja zapis WSDL (Web Service Description Language).

3.3.4 XML

XML je kratica za angleški izraz Extensible Markup Language, ki bi ga lahko v slovenščino prevedli kot razširljiv označevalni jezik. Omogoča nam zapis strukturiranih podatkov. Njegova razširljivost temelji na možnosti, da uporabnik sam določi elemente, ki jih bo uporabljal. Najpogosteje se ta zapis uporablja za prenos preko omrežja.

Zapis podatkov v obliki XML priporoča tudi konzorcij W3C, ki predstavlja vodilno mednarodno organizacijo za postavitve standardov na svetovnem spletu.

3.3.5 Microsoft Visual Studio

Microsoft Visual Studio (VS) predstavlja integrirano razvojno okolje proizvajalca programske opreme Microsoft. Omogoča razvoj v več različnih programskih jezikih kot tudi različne tipe aplikacij. Med najpomembnejšimi funkcionalnostmi okolja so podpora naprednemu razhroščevanju, različni grafični urejevalniki in čarovniki, podpora skupinskemu delo, itd..

3.3.6 XAML

XAML (Extensible Application Markup Language) predstavlja deklarativen jezik, ki ga pri Microsoftu uporabljajo za zapisovanje struktur in vrednosti. Svoj razmah je zapis doživel s prihodom .NET Framework 3.0, kjer se lahko uporablja za zapis uporabniškega vmesnika in delovnih tokov v programskem modelu Windows Workflow Foundation.

Datoteke, ki opisujejo delovne tokove, imajo končnico .XOML. Razlog za to je zgodovinski, ker se je včasih zapis uporabniškega vmesnika ali delovnega toka razlikoval. Te razlike ni več.

3.3.7 ODBC

Open Database Connectivity ali s kratico ODBC predstavlja standardizirani vmesnik za dostop do sistema za upravljanje s podatkovnimi bazami. Namen takšnega vmesnika je bil omogočiti programerju razvoj aplikacij neodvisno od specifik podatkovnih baz.

3.3.8 Windows Service

Windows Service predstavlja dolgo trajajoč proces, ki izvaja določeno funkcionalnost in ne potrebuje posredovanja uporabnika. Zato le ta nima uporabniškega vmesnika. Lahko ga poženemo ročno, ali pa se požene sam pri zagonu operacijskega sistema.

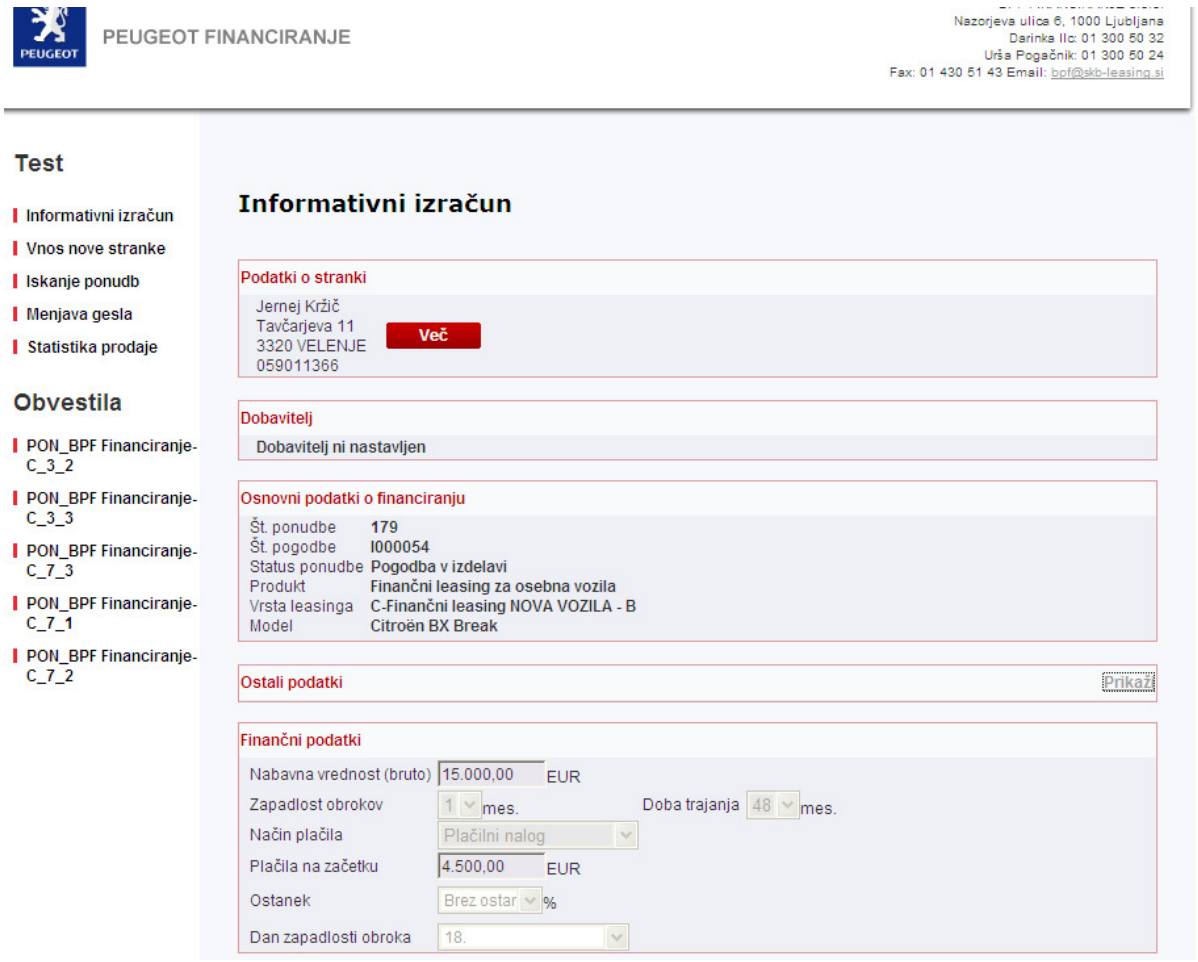
3.4 Produkt AdLeasing

3.4.1 Osnovna predstavitev

Produkt AdLeasing je informacijski sistem za podporo prodaje leasing produktov. Izdelan je kot spletna aplikacija, razvita v tehnologiji ASP.NET 3.5. Ta nam omogoča razvoj aplikacije, ki ne potrebuje namestitve pri vsakem uporabniku posebej, ampak jo namestimo na spletnem strežniku. Uporabniki za uporabo aplikacije potrebujejo le spletni brskalnik.

Postopek se prične z informativno ponudbo, ki jo pripravi pooblaščeni posrednik. Pri tem stranki pripravi informativne izračune in simulacije, ki prikažejo mesečne obveznosti in efektivne obrestne mere pri različnih dobah odplačevanja. Če se stranka za katero izmed ponudb odloči, gre ponudba naprej k pooblaščenemu uslužbencu leasingodajalca. Temu se poleg vseh parametrov ponudbe prikažejo tudi računalniška ocena tveganja posla. Ta se izračuna glede na vnesene podatke stranke in parametre ponudbe.

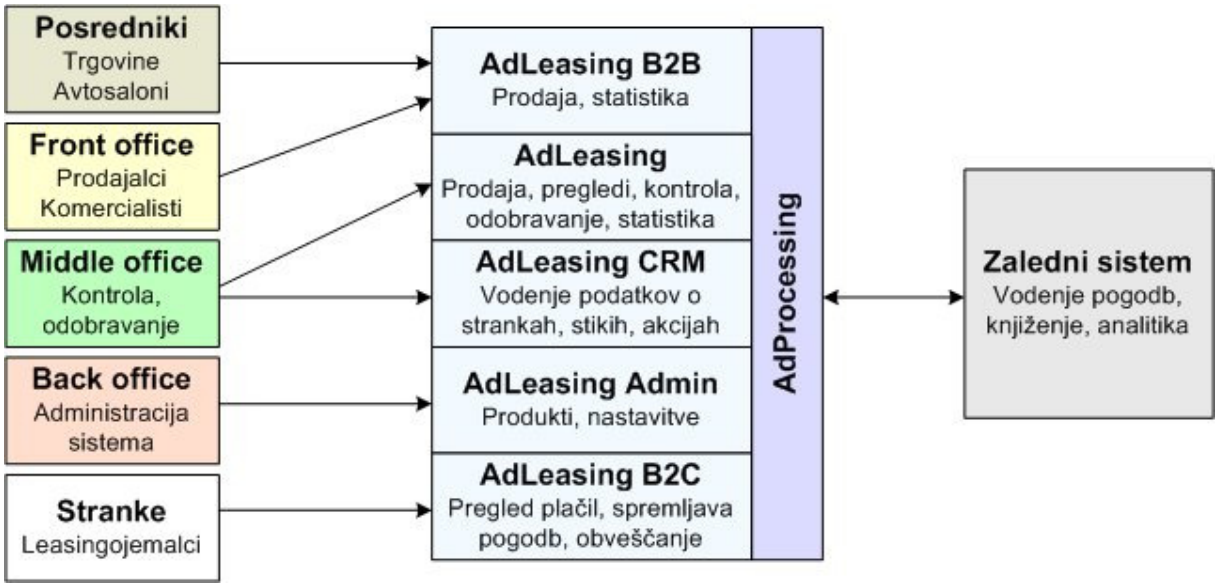
Ocena služi zaposlenemu le kot usmeritev, ta pa ima možnost podpisati ali pa jo zavrniti. Ko se na ponudbi zbere dovolj podpisov, je ponudba odobrena. Število potrebnih podpisov je odvisna od višine financirane vrednosti, tipa financiranja in računalniške ocene stranke. V tem statusu se natisne različna dokumentacija, potrebna za izvedbo posla. Po potrditvi pa se ponudba prenese v zaledni sistem, kjer skrbijo za redno poravnavanje obrokov in v primeru sprememb referenčne obrestne mere izvajajo ponovne obračune plačil. Dnevno se vsi ti podatki prenašajo nazaj v AdLeasing, kjer so na voljo strankam preko B2C modula. Tako lahko stranke spremljajo stanje pogodb, datume zapadlosti obrokov, podajajo reklamacije in postavljajo vprašanja.



Slika 12: Vnos osnovnih podatkov o financiranju

Slika 13: Prikaz izračunanih parametrov financiranja in podatkov o oceni bonitete stranke

3.4.2 Moduli rešitve AdLeasing



Slika 14: Moduli rešitve AdLeasing

Produkt AdLeasing ima modularno zgradbo. Tako smo nekatere module implementirali pri vseh naročniki, nekateri pa so izdelani za točno določenega. Tako je modul za prodajo, statistiko in administracijo enak za vse. Modul za odobravanje pa je razvit za vsako implementacijo posebej. To niti ne preseneča, saj je znanje, uporabljeno za izdelavo tega modula, strogo varovana skrivnost vsake leasing hiše posebej. Nekaj prilagoditev je zahteval tudi modul za vnos in ogled podatkov leasingojemalcev. Modul za podporo delu s strankami CRM pa je bil razvit posebej za VBS Leasing.

4 Obravnavani problem

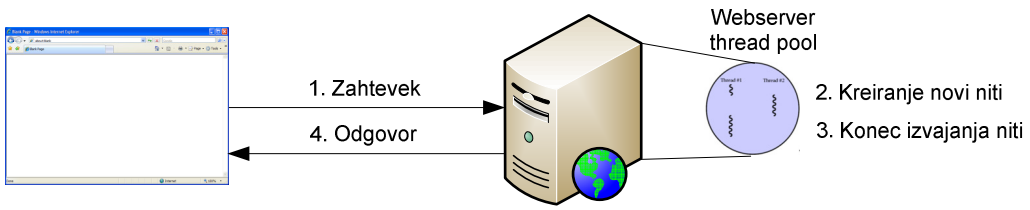
4.1 Cilj uvedbe WF v obstoječo aplikacijo

Osnovni cilj te naloge je uporabiti že izdelano, delujočo aplikacijo in vanjo vpeljati programski model Windows Workflow Foundation. Pri tem bomo zasledovali sledeče spremembe na obstoječi aplikaciji:

- zmanjšanje količine izvorne kode na vseh plasteh aplikacije,
- olajšano vzdrževanje aplikacije,
- omogočeno vključevanje poznavalcev vsebinskega področja, ki ga aplikacija pokriva,s čimer se zmanjša semantični prepad med naročnikom in izvajalcem,
- z razvojem lastnih aktivnosti zagotoviti preprostejši razvoj in vzdrževanje WF,
- pospešiti razvoj določenih delov aplikacije.

4.2 Posebnosti uporabe WF v ASP.NET aplikaciji

Izvajalno okolje Windows Workflow Fondation se izvaja na posebni niti znotraj gostujočega procesa. Takšen način mu omogoča asinhrono izvajanje delovnih tokov neodvisno od gostujoče aplikacije. Tako lahko gostujoča aplikacija delovni tok požene in nadaljuje s svojim delom. Delovni tok pa jo nato s pomočjo dogodkov obvešča, da potrebuje dodatne parametre, odločitve uporabnika ali pa je zaključil z izvajanjem in ima na voljo rezultat. Še pomembnejše pa je, da zna izvajalno okolje v primeru časovnih zamikov izvajanja samo ugotoviti, kdaj je ta zamik potekel in se mora izvajanje toka nadaljevati. Ravno s posebno nitjo, ki neprestano bedi nad temi zamiki, WF zagotavlja, da se bo izvajanje nadaljevalo v predvidenem časovnem zamiku.



Slika 15: Izvajanje spletne aplikacije

ASP.NET aplikacija pa ima nekoliko specifičen potek izvajanja. Deluje po principu zahtevek/ odgovor (ang. Request/Response), kot to kaže slika. V času med prejšnjim odgovorom in novo zahtevo se aplikacija ne izvaja. Ponovno se prične izvajati, ko na strežnik prispe naslednja zahteva. Uporaba WF izvajalnega okolja povzroči, da vsaka zahteva odpre dve izvajalski niti, prva nit za izvajanje ASP.NET aplikacije in druga za izvajanje delovnih tokov. V internetnem svetu se na enem strežniku hkrati izvaja več zahtevkov, število hkratnih niti pa je omejeno. Tako naletimo na težavo pri gostovanju WF znotraj spletne aplikacije.

Dani zadregi se deloma izognemo z uporabo posebne WF storitve, ki omogoča izvajanje WF okolja na isti niti kot ASP.NET aplikacijo. Pri tem pa izgubimo asinhrono delovanje in ostale prednosti, ki sem jih opisal na začetku tega poglavja.

4.3 Zagotavljanje sledljivosti

Na delovni tok lahko gledamo tudi kot na črno škatlo, v katero pošljemo začetne parametre. Med njegovim delovanjem pa se odzivamo na njegove zahteve. Na koncu pa preberemo njegov rezultat. Kljub temu pa kmalu pridemo tudi do potrebe po pojasnitvi, kako je tok potekal, kako se je na razpotjih odločal in kako je prišel do rezultata. Brez tega je uporaba delovnih tokov v kritičnih delih neuporabna.

Windows Workflow Foundation omogoča sledljivost s pomočjo uporabe tako imenovanih storitev, ki jih poljubno dodajamo okolju za izvajanje delovnega toka. Že v osnovni namestitvi imamo podporo za razvoj lastne storitve beleženja poteka. Uporabimo pa lahko že narejeno, ki za hrambo sledi uporablja SQL strežnik.

Po vključitvi storitve beleženja se v izbran SQL strežnik samodejno beležijo podatki o zaporedju izvajanja aktivnosti. Le ti zadostujejo, da po končanem izvajanju skupaj z definicijo delovnega toka tudi grafično prikažemo sled izvajanja.

4.4 Zagotavljanje vzdrževanja različic

Zelo pomembno poglavje, ki ga večina literature o Windows Workflow Foundation preprosto preskoči, je skrb za različice tokov. Aplikacije so dinamičen sistem, ki se redno spreminja, posodablja in razvija.

Spremembe pa lahko delimo v dve skupini:

- prilagoditev delovnega toka zaradi spremembe poslovnih pravil,
- prilagoditve, ki veljajo tudi za obstoječe delovne tokove.

Težave nam sicer povzročajo le tokovi, ki se izvajajo dalj časa in se med čakanjem na uporabnikov odziv shranijo na obstojni podatkovni nosilec. V kolikor zamenjamo le zbir z novo različico, se obstoječi tokovi ne bodo več obnovili in nadaljevali z delom.

4.4.1 Prilagoditve zaradi spremembe poslovnih pravil

Pri spremembi toka zaradi spremembe poslovnih pravil je rešitev preprostejša. Že samo ogrodje .NET Framework vsebuje podporo različicam zbirov, razvijalci se moramo le držati splošnih načel le tega. Na tak način lahko dosežemo, da se delovni tok **izvaja in zaključi z upoštevanjem pravil, s katerimi se je tudi pričel.**

Najprej moramo poskrbeti za digitalno podpisovanje zbira. S pomočjo javnega ključa, ki se doda zbiru pri digitalnem podpisovanju, lahko izvajalno okolje .NET Framework razlikuje različne zbire. Tudi če vsebujejo enake imenske prostore, razrede in metode. Potrebno je tudi skrbeti za konsistentno spreminjanje različice zbira. Ob upoštevanju obeh omenjenih postopkov bomo lahko imeli nameščenih več različic zbira. Govorimo o tako imenovani »side by side« namestitvi.

Kadar uporabljamo različice in želimo, da izvajalno okolje samo izbere pravo, bomo zbire dodajali v univerzalno shrambo zbirov (angleško Global Assembly Cache oz. GAC). Zbire, ki so hranjeni v GAC-u, lahko uporablja vsaka .NET aplikacija. Vsebovati pa mora podatek o različici in mora biti digitalno podpisan.

Namestitev zbira v GAC se izvede s pomočjo orodja »gacutil.exe« in je del namestitve .NET Frameworka. Uporaba GAC-a nam nekoliko oteži razširjanje naše aplikacije, saj ne moremo le te samo prekopirati in uporabljati, temveč moramo še prej poskrbeti za namestitev izbranih zbirov v GAC.

4.4.2 Prilagoditve, ki veljajo tudi za obstoječe delovne tokove

Na večjo težavo pa naletimo, v kolikor želimo spremeniti tudi delovanje delovnih tokov, ki se že izvajajo. To moramo storiti v primeru, da smo pri razvoju delovnega toka naredili napako, ki le temu preprečuje nadaljevanje izvajanja. Obstaja možnost predhodnega končanja trenutnega toka in pričetek novega z novo različico pravil. Ker pa lahko primerek toka traja precej dolgo in gre skozi veliko faz, ki zahtevajo odziv uporabnika, ta rešitev pogosto ne pride v poštev. V tem primeru si lahko pomagamo z uporabo dinamičnih posodobitev (ang. Dynamic Updates). Žal pa je takšno prilagajanje precej kompliciranо in se ga izplača uporabiti le v izjemnih primerih pri minimalnih korekcijah pravil. Z uporabo dinamičnih

posodobitev lahko posegamo v zgradbo točno določenega primerka delovnega toka. Na takšen način lahko dodajamo nove aktivnosti, spreminjamo nastavitve in brišemo obstoječe.

Spremembe lahko izvajamo interno, se pravi znotraj delovnega toka. Z drugimi besedami, tako, da delovni tok sam prilagaja svoje delovanje. V kolikor pa posodobitve izvajamo zunaj njega, pa moramo paziti na nekatere omejitve. Ena izmed njih je, da se tok v času sprememb ne sme izvajati. Prav tako ima delovni tok možnost, da se prepreči spremembe. Pri tem je potrebno poudariti, da se spremembe ne uveljavljajo neposredno, ampak se po zaključku urejanja izvede še preverjanje veljavnosti sprememb. Če je to uspešno, se spremembe uveljavijo, v nasprotnem primeru nam izvajalno okolje vrne seznam napak. Pri tem bi rad poudaril, da se spremembe uveljavijo le na posameznemu primerku delovnega toka.

Na različnih internetnih forumih potekajo diskusije, kako spreminjati delovanje že pričetih delovnih tokov. Nekateri izmed njih vsebujejo tudi neposredne spremembe podatkov v podatkovni bazi, ki jo WF uporablja za shranjevanje stanja delovnega toka. Na tem mestu bi rad opozoril, da to niso uradno podprte rešitve Microsofta in da lahko takšno poseganje privede do nepredvidljivega delovanja WF, ki pa se lahko pokaže šele čez čas.

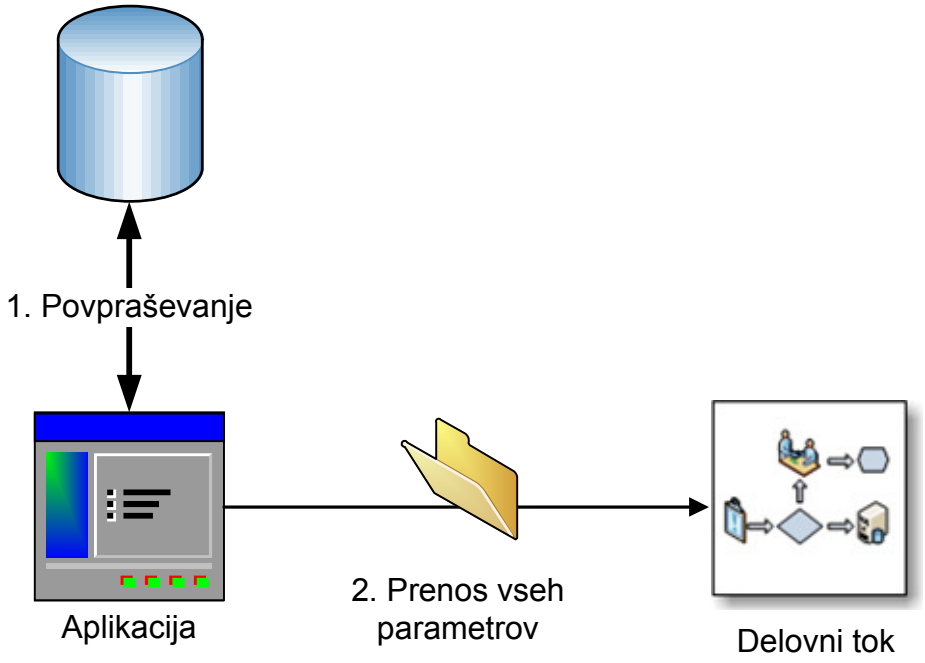
4.5 Prenosi parametrov za delovanje delovnega toka

Delovni tok za svoje izvajanje in odločanje potrebuje podatke. Le te mu lahko posredujemo na več načinov, in sicer:

- kot vhodni parameter pri zagonu delovnega toka,
- kot povpraševanje, ki ga sproži delovni tok, nanj pa odgovori gostujoča aplikacija,
- povpraševanje delovnega toka v podatkovni bazi,
- črpanje podatkov preko poljubnega standardiziranega podatkovnega vmesnika.

Prednosti in omejitve naštetih možnosti pa opisujem v naslednjih poglavjih.

4.5.1 Kot vhodni parametri



Slika 16: Prikaz prenosa podatkov kot vhodni parametri

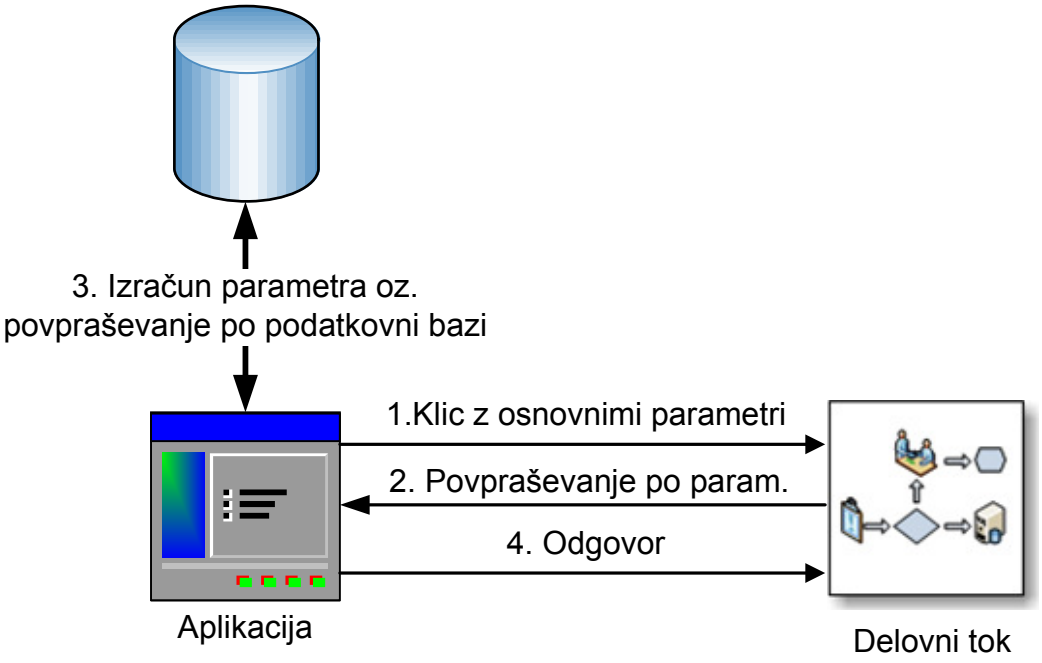
V tem primeru vse potrebne parametre podamo že na začetku izvajanja delovnega toka. Prednost takšnega pristopa je predvsem preprostost uporabe.

Vendar pa to zahteva podajanje vseh podatkov ne glede na to, ali jih bo delovni tok potreboval ali ne. Še večji problem pa nastane pri dolgo trajajočih delovnih tokovih. Saj so se lahko ti parametri v času izvajanja spremenili, kar vodi v nekonsistentno stanje delovnega toka.

Na še večji problem pa naletimo, ko želimo spremeniti delovanje delovnega toka in zahtevati še kakšen dodaten podatek. Takrat moramo istočasno popraviti tudi vse aplikacije, ki ta delovni tok uporabljajo.

Tako tega pristopa ne bomo uporabljali, razen kadar gre za majhen nabor parametrov, ki se ne spreminja. V nasprotnem primeru bomo uporabili enega izmed naprednejših načinov.

4.5.2 Dodatno povpraševanje



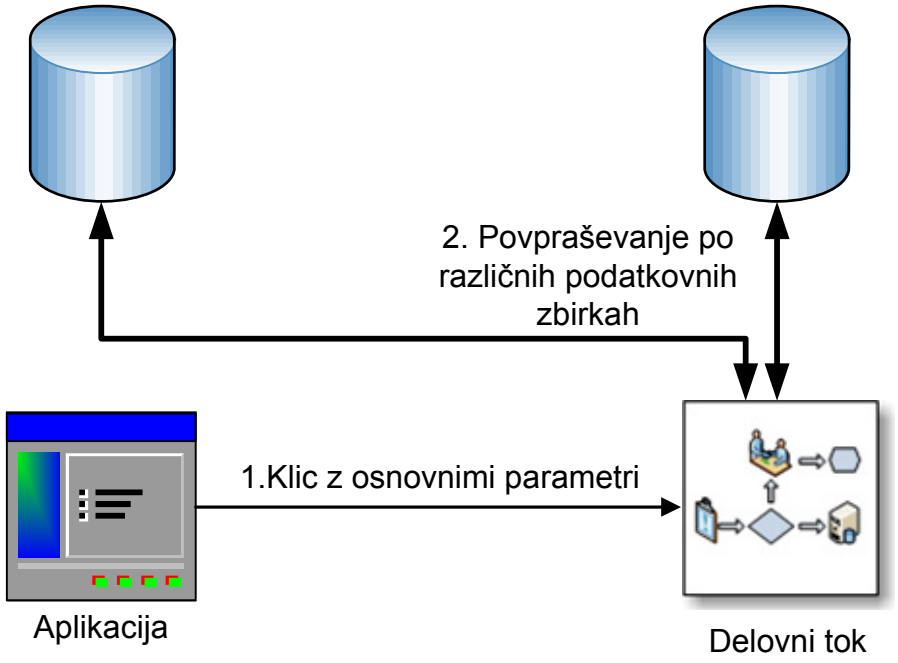
Slika 17: Prikaz prenosa podatkov kot dodatno povpraševanje

Windows Workflow Foundation s pomočjo tako imenovanih Local Services omogoča tudi povratno komunikacijo. Tako lahko delovni tok od aplikacije zahteva dodatne parametre.

Takšen način komunikacije je predvsem priročen, ker delovni tok sam ugotavlja, katere parametre v danem trenutku potrebuje. Tako ima tok tudi trenutno aktualne podatke, ki so pogoj za konsistentno delovanje. Dodatna pozitivna lastnost takšnega pristopa je vmesna stopnja med delovnim tokom in shrambo podatkov. Tako se aplikacija sama odloča, kje so podatki shranjeni in lahko izvrši obdelavo podatkov, preden jih posreduje naprej.

Slabost takšnega pristopa je spreminjanje nabora možnih parametrov. V tem primeru moramo istočasno popraviti vse aplikacije, ki delovni tok uporabljajo. Prav tako so omejitve, v katerem primeru takšno povpraševanje lahko izvajamo. Več o teh omejitvah je napisano v poglavju 4.6.

4.5.3 Povezava na podatkovno bazo



Slika 18: Prikaz prenosa podatkov kot povpraševanje neposredno na podatkovni bazi

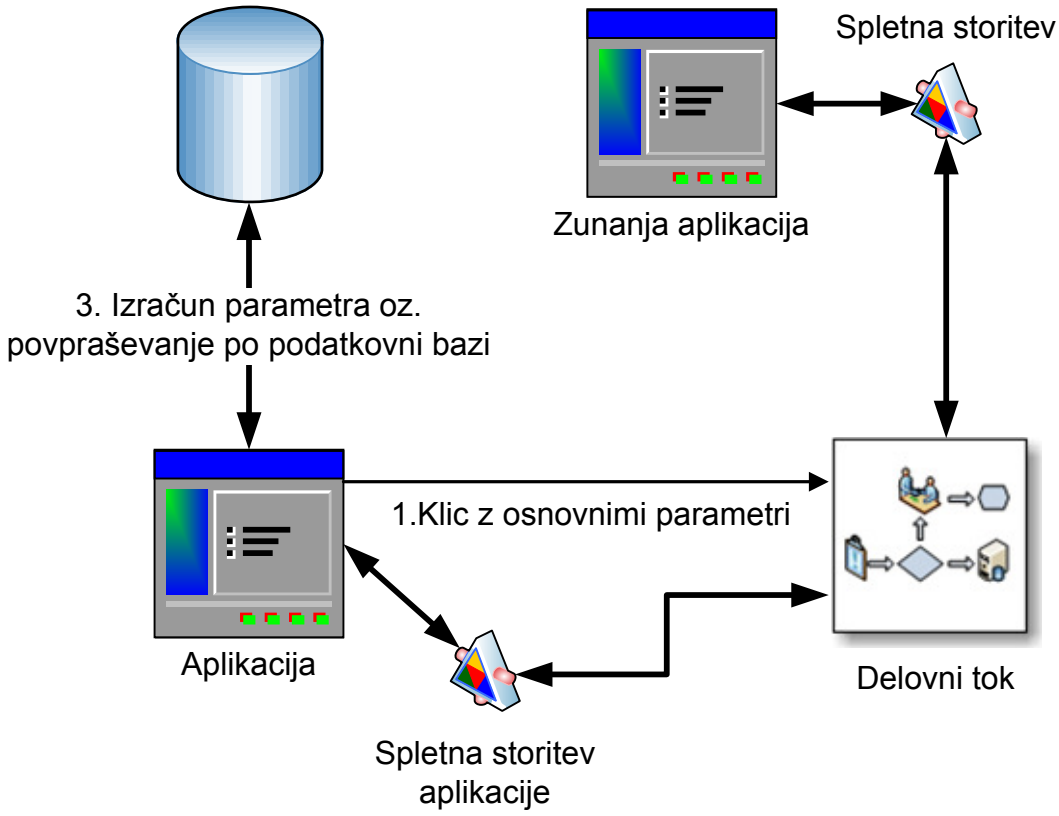
Naslednja možnost za pridobivanje podatkov je neposredna povezava na podatkovno bazo. Na takšen način lahko delovni tok pridobiva le nujno potrebne podatke in to v trenutku, ko jih potrebuje za odločitev. Slednje zagotavlja, da vedno dela s konsistentnimi podatki.

Ker povpraševanja izvaja WF sam, aplikaciji, ki uporablja storitve delovnega toka, ni potrebno vedeti, katere podatke delovni tok potrebuje in kje jih lahko pridobi. Tako lahko v resnici govorimo o ponovno uporabljivih delovnih tokovih.

Windows Workflow Foundation nima vgrajene podpore za neposredno povpraševanje po podatkovnih bazah. Ker pa omogoča pisanje lastnih aktivnosti, smo le to razvili sami. Te aktivnosti nam omogočajo povezovanje preko vmesnika ODBC, s katerim imamo dostop do večine podatkovnih baz.

Slabost takšnega pristopa je neposredno povpraševanje in celo spreminjanje podatkov mimo poslovne logike aplikacije. To lahko vodi v nekonsistentno stanje podatkov in posredno v nepredvidljivo delovanje aplikacije. Zato se takšnega pristopa poslužujemo le za preprostejše poizvedbe.

4.5.4 Črpanje podatkov preko standardiziranega vmesnika



Slika 19: Prikaz prenosa podatkov preko standardiziranih vmesnikov

Če združimo pozitivne lastnosti dodatnega povpraševanja, opisanega v poglavju 4.5.2, in možnost samodejnega pridobivanja podatkov iz podatkovne baze, o katerem govorim v poglavju 4.5.3, dobimo najnaprednejši način za pridobivanje podatkov.

Tako lahko delovni tok povsem samostojno pridobiva podatke iz več različnih aplikacij, ki so opremljene s standardiziranim podatkovnim vmesnikom. Ko dandanes govorimo o standardiziranih vmesnikih, v večini mislimo na spletne storitve. Tako nam bo v pomoč tudi že pripravljena aktivnost za uporabo le teh.

Ker delovni tok nima direktnega dostopa do podatkovne baze, pa to pripomore tudi k varnosti celotne rešitve. Tako gredo vsa povpraševanja še preko poslovne logike same aplikacije, kjer se lahko izvaja dodatno preverjanje pravil za dostop do podatkov.

Poglavitna slabost takšnega pristopa je predvsem dodatna potreba po sistemskih sredstvih, ki jo prinesejo klici spletnih storitev. Tu mislim predvsem na pretvarjanje podatkov v standardizirano obliko XML.

4.5.5 Primerjava prenosov parametrov

	Možnost uporabe iz več različnih aplikacij	Prenos le nujno potrebnih podatkov za izvedbo delovnega toka	Dodatna vmesna stopnja med tokom in podatki za izvajanje poslovne logike	Potreben razvoj lastnih WF aktivnosti
Kot vhodni parametri	DA, z omejitvami*	NE	DA	NE
Dodatno povpraševanje	DA, z omejitvami*	DA	DA	NE
Povezava na podatkovno bazo	DA	DA	NE	DA
Črpanje podatkov preko standardiziranega vmesnika	DA	DA	DA	NE

* - pri spremembi nabora potrebnih podatkov za izvedbo delovnega toka moramo istočasno popraviti delovanje VSEH aplikacij, ki koristijo storitve delovnega toka

4.6 Izbor načina uporabe WF v obstoječi aplikaciji

Kot sem podrobno opisal že v poglavju 3.2.8, lahko Windows Workflow Foundation v našo obstoječo aplikacijo vključimo na več načinov. Vsak izmed njih ponuja svoje prednosti in prinaša nekatere slabosti. Tako se bomo morali za vsak primer posebej odločiti, kateri izmed njih je v danem primeru primernejši. Odločali se bomo po sledečih parametrih:

- kako pogosto poganjamo delovne tokovi in kakšno odzivnost zahtevamo,
- ali uporabljamo kratkotrajne ali dolgo trajajoče delovne tokove,
- koliko parametrov delovni tok potrebuje za izvršitev in na kakšen način jih bomo posredovali v delovni tok,
- ali imamo navadno odjemalsko aplikacijo, strežniško ali spletno aplikacijo,
- ali delovni tok zahteva posredovanje uporabnika,
- ali bomo delovni tok klicali iz ene ali z več različnimi aplikacijami,
- v katerem ogrodju je obstoječa aplikacija napisana.

4.6.1 Gostovanje znotraj aplikacije

Predpogoj za možnosti gostovanja WF v aplikaciji sami je, da je aplikacija razvita v .NET združljivem jeziku in da uporablja vsaj ogrodje različice 3.0 ali novejše. Glede na to, da poslovne aplikacije razvijamo dalj časa, prav izbira trenutnega ogrodja predstavlja omejitev. Gostovanje delovnih tokov znotraj ASP.NET spletne aplikacije je sicer možno, vendar z nekaterimi omejitvami, opisanimi v poglavju 4.2.

Gostovanje WF izvajalnega okolja znotraj aplikacije pa prinaša tudi več pomembnih prednosti. Najpomembnejša je ta, da le na takšen način res lahko izrabimo vse zmožnosti, ki jih WF omogoča. Na takšen način bomo brez težav gostovali hitre in odzivne kratkotrajne delovne tokove, kot tudi dolgo trajajoče z možnostjo vmesnega hranjenja. Brez težav bomo omogočili tudi interakcijo z uporabnikom kot tudi s samo aplikacijo.

Takšen način uporabe WF je najprimernejši za odjemalske Windows aplikacije, ki potrebujejo hitro in odzivno uporabo storitev WF z vsemi možnostmi, ki jih le ta ponuja. Razvite morajo biti v združljivem ogrodju.

4.6.2 Objava kot spletna storitev

Zelo priročen način, da storitve delovnega toka ponudimo širšemu krogu aplikacij, je objava le tega kot spletna storitev. Za to najprej potrebujemo spletni strežnik IIS ali katerega drugega, ki omogoča gostovanje ogrodja .NET Framework.

Za gostovanje poskrbi ASP.NET in spletni strežnik. Le to pa prinese nekatere specifike, ki smo jih vajeni iz spletnih aplikacij. Te namreč delujejo po principu zahtevek / odgovor. In tako so zaporedni zahtevki načeloma neodvisni. Ker pa je lahko delovni tok sestavljen iz več zahtevkov in odgovorov, je potrebno le te povezati. Za to uporabljamo piškotke (ang. cookie), ki omogočajo hranjenje omejene količine podatkov pri uporabniku spletne aplikacije in se posredujejo skupaj z ostalimi parametri ob vsaki zahtevi. Ostale značilnosti gostovanja delovnih tokov z ASP.NET-om sem opisal v poglavju 4.2.

Pomembna omejitev takšnega načina uporabe je nezmožnost povratne komunikacije. Tako lahko delovni tok le vrača odgovore na zahtevke. Ne more pa sam poslati dodatnega podatka, ki ga potrebuje za odločitev.

4.6.3 Uporaba WCF

Windows Communication Foundation je programski model, ki omogoča gradnjo aplikacij, ki zanjo komunicirati z drugimi sistemi.

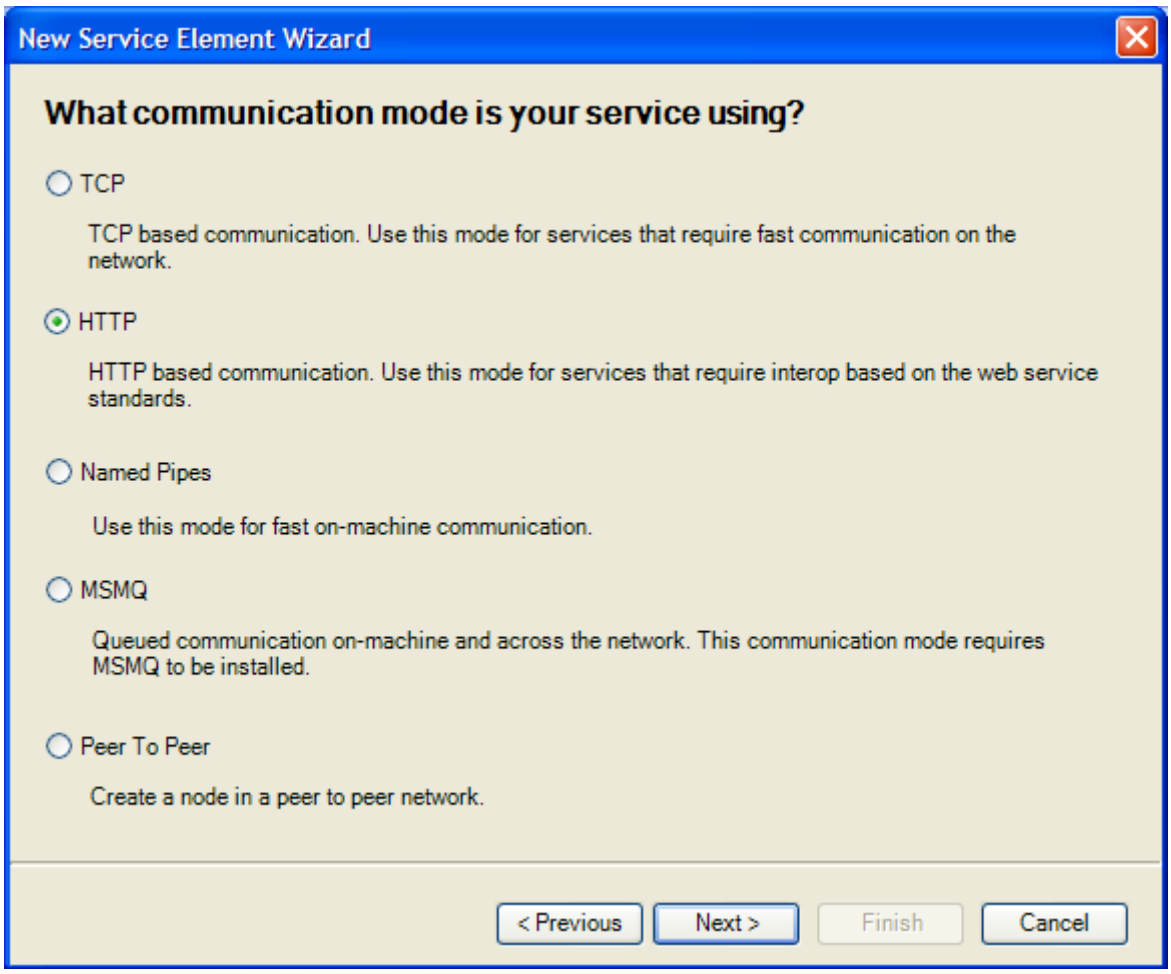
Sam WCF tako ne more samostojno gostovati delovnih tokov, zato bomo vseeno potrebovali aplikacijo. Ta je lahko v obliki Windows aplikacije ali kot sistemski Windows Service.

Prvi korak je razvoj razširitve za WCF razširitve (ang. WCF extension). Le ta poskrbi za zagon WF izvajalnega okolja, konfiguracijo le tega in po zaključku dela tudi ustavitev okolja in sprostitev zaseženih sistemskih sredstev.

Naslednji korak je razvoj lastne storitve. Pri tem moramo določiti storitveno pogodbo. Ta določa operacije, ki jih naša storitev izvaja. Po potrebi pa določimo tudi podatkovno pogodbo, ki služi za opis morebitnih kompleksnejših podatkovnih tipov. Bolj podrobno sem WCF opisal v poglavju 3.2.6.

V tretjem koraku povežemo prva dva. To storimo tako, da najprej kreiramo novo aplikacijo. V njej odpremo novega gostitelja WCF za našo storitev iz koraka dve. Ne smemo pa pozabiti dodati še razširitve iz prvega koraka. V nasprotnem primeru naša storitev ne bo imela dostopa do izvajalnega okolja WF.

Tako dobimo WF, ki ga lahko uporabljajo različne aplikacije. Za delovanje ne potrebujemo dodatne programske opreme, kot je recimo za spletne storitve potreben spletni strežnik. Pomembna prednost je tudi možnost spreminjanja načina delovanja naše storitve. Le te se določajo preko nastavitvene datoteke v obliki XML, brez zahteve po ponovnem prevajanju aplikacije. Takšen pristop omogoča več načinov povezovanja, med njimi TCP, spletne storitve, sporočilne vrste, ipd.….



Slika 20: Načini povezovanje z WCF

Z uporabo WCF lahko obidemo tudi omejitev spletnih storitev, ki omogočajo le komunikacijo zahtevek / odgovor. WCF namreč omogoča vzpostavitev dvosmernega (ang. duplex) kanala.

4.6.4 Primerjava primerov uporabe WF

	Uporaba storitev WF v poljubni aplikaciji	Potrebna namestitev dodatne programske opreme	Možnost dvosmerne komunikacije med WF in aplikacijo
Gostovaje znotraj aplikacije	NE	NE	DA
Objava kot spletna storitev	DA	DA	NE
Uporaba WCF	DA*	NE	DA

* - WF omogoča več načinov povezovanja. Za uporabo storitev WF v poljubni aplikaciji moramo uporabiti standardizirane načine, kot so na primer spletne storitve.

5 Uvedba Windows Workflow foundation v produkt AdLeasing

5.1 Izdelava in uporaba podpornih aplikacij

Ogrodje WF vsebuje veliko funkcionalnosti »out of the box«. Tako v računalništvu radi poimenujemo nabor funkcionalnosti, ki jih ponuja produkt brez namestitve drugih produktov, dodatkov ali nadgradenj. Samo ogrodje pa ponuja tudi veliko možnosti, da vanj vključujemo delčke svojih rešitev ali prilagoditev. Le te nam omogočajo, da določene funkcionalnosti razvijemo samo enkrat in jih uporabljamo na več različnih mestih. Takšen pristop pohitri razvoj in olajša vzdrževanje aplikacij.

Pri uvedbi WF v produkt AdLeasing smo razvili nekaj standardiziranih aktivnosti kot tudi nekaj dodatkov, potrebnih za podporo BPM.

5.1.1 Razvoj lastnih aktivnosti

Pri uvedbi WF v produkt AdLeasing sem pogrešal nekatere funkcionalnosti. Zato sem jih razvil sam v obliki lastnih aktivnosti. Tako jih bo mogoče uporabljati tudi na drugih projektih, kjer bomo uporabljali WF. To sta:

- aktivnost za neposreden dostop do podatkovne baze,
- aktivnost za pošiljanje email sporočil.

5.1.1.1 Aktivnost za neposreden dostop do podatkovne baze

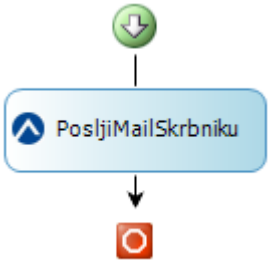
Ta nam omogoča, da se povezujemo na poljubno podatkovno bazo, za katero obstaja ODBC združljivi gonilnik. ODBC predstavlja standardizirani vmesnik med aplikacijo in podatkovno bazo, kar zagotavlja neodvisnost delovnega toka od podatkovne baze.

Namen razvoja takšne aktivnosti je bil, omogočiti delovnemu toku, da sam pridobiva potrebne parametre direktno iz podatkovne baze. Na takšen način si zagotovi neodvisno delovanje in konsistentne podatke.

5.1.1.2 Aktivnost za pošiljanje email sporočil

Kljub poplavi sistemov za podporo skupinskemu delu, kot je Microsoft SharePoint in podobni, se v praksi izkaže kot najučinkovitejši način obveščanja uporabnikov ravno uporaba email (ali celo SMS) sporočil. Tako se mi je zdela zelo uporabna ravno aktivnost za pošiljanje elektronskih sporočil.

Glede na to, da ima ogrodje .NET Framework solidno podporo za pošiljanje elektronskih sporočil, razvoj takšne aktivnosti ni bila ravno težka naloga. Ker pa je izvedena v obliki aktivnosti v lastni knjižnici, pa bo to rešitev moč uporabljati tudi v drugih podobnih projektih.

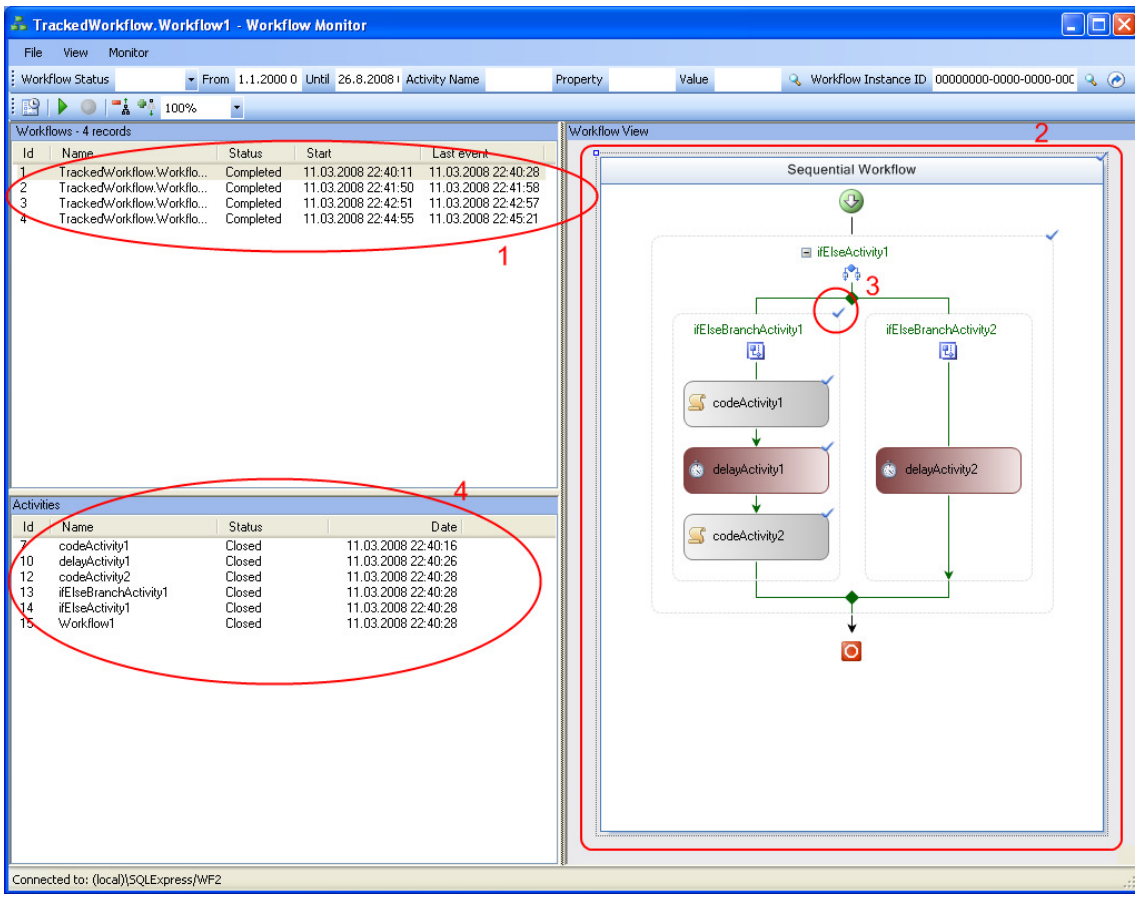


Slika 21: Primer uporabe aktivnosti za pošiljanje email sporočil

5.1.2 Grafični prikaz poteka delovnega toka

Upravljanje s poslovnimi procesi od nas zahteva tudi neprestan nadzor in spremljanje poslovnih procesov. V ta namen uporabljamo posebno aplikacijo. Le ta nam omogoča grafičen prikaz delovnega toka in pot, po kateri se je le ta izvajal.

Kot osnovo sem vzel aplikacijo Workflow Monitor, ki je del primerov programov iz WF. Dopolnil pa sem jo tako, da sedaj prikazuje še nekatere dodatne podatke delovnega toka, ki so pomembne za nadzor in spremljanje izvajanja. Tako lahko vidimo za vsako aktivnost, kdaj se je pričela in kdaj končala. Tako lahko prepoznamo ozka grla in dele procesa, kjer je možna še dodatna optimizacija.



Slika 22: Grafični prikaz poteka delovnega toka

Aplikacija tako vsebuje seznam delovnih tokov, ki so pričeli z izvajanjem. Le ta je označen s točko 1. Ta seznam lahko dodano omejimo z uporabo dodatnih filtrov v orodni vrstici aplikacije. Poleg imena delovnega toka vidimo še njegov status ter datum in čas pričetka izvajanja in zadnjega dogodka. Če izberemo katerega izmed njih, se nam v oknu, označenem s točko dve, prikaže delovni tok. Pri tem lahko vidimo imena vseh aktivnosti, s kljukico (glej točko 3) pa so označene aktivnosti, ki so bile izvedene.

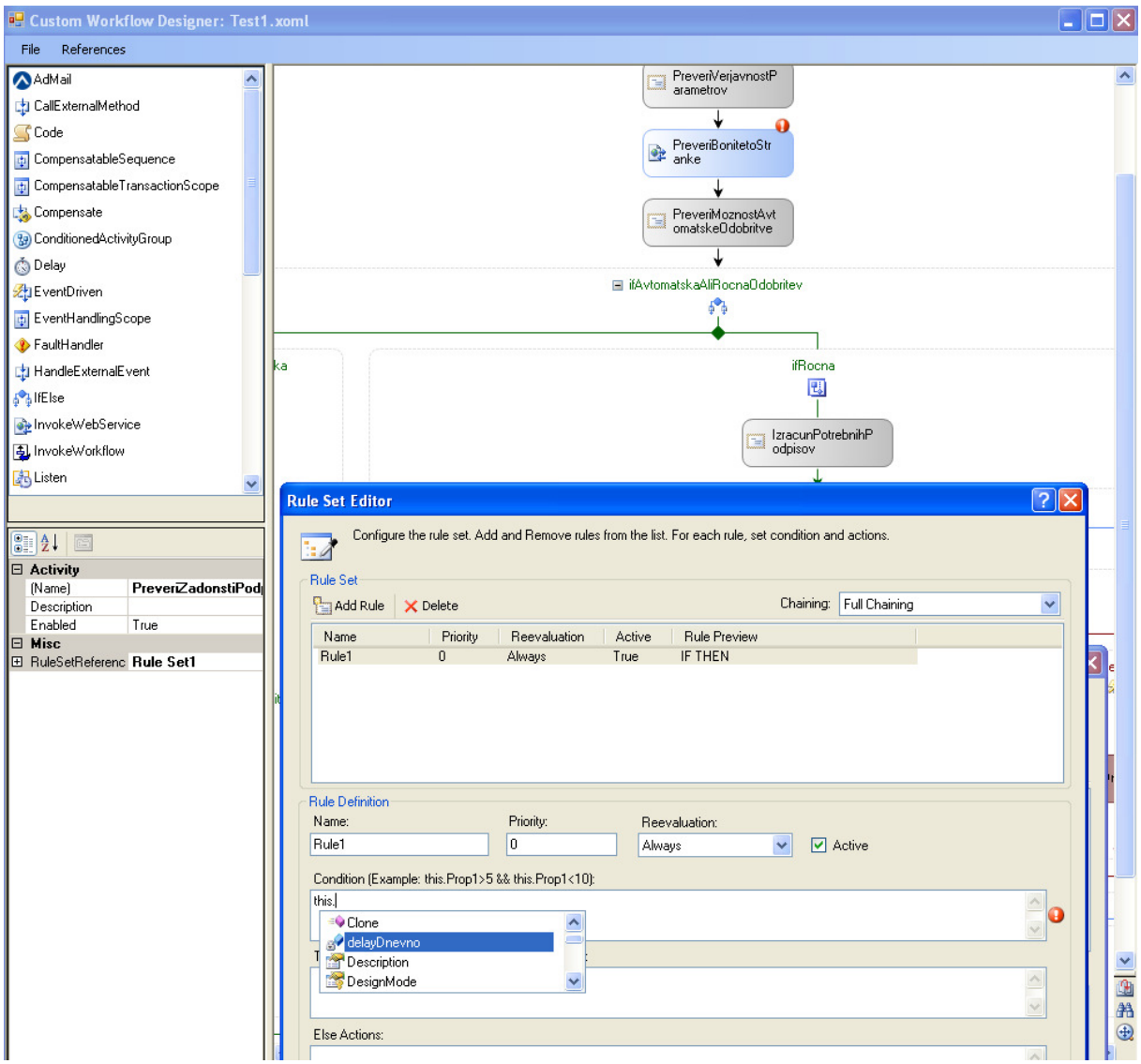
V oknu, označenem s točko 4, je viden seznam vseh aktivnosti, ki so že pričele z izvajanjem. V seznamu je vidno ime aktivnosti (enako kot tudi na grafičnem prikazu toka), njen status in datum pričetka izvajanja. Ravno s primerjavo časov izvajanja določenih aktivnosti lahko prepoznamo tiste, ki zavirajo celoten poslovni proces. V danem primeru sta počasni le aktivnosti »codeActivity1« in »delayActivity1«. Ostale se izvedejo tako rekoč v trenutku in njihova optimizacija ne bi bila smiselna.

5.1.3 Samostojen urejevalnik podatkovnih tokov

Windows Workflow foundation s svojo modularno zgradbo omogoča dokaj preprosto vključitev standardiziranega urejevalnika v našo lastno aplikacijo. Urejevalnik je povsem enak kot tisti znotraj Microsoftovega razvojnega orodja Visual Studio. Izgled in delovanje urejevalnika sem opisal v poglavju 3.2.4.

Velika prednost takšne rešitve je v tem, da želimo skozi grafičen prikaz delovnih tokov v njihovo izgradnjo in vzdrževanje vključiti tudi uporabnike naših aplikacij. Le ti najbolje vedo, kako njihovi poslovni procesi potekajo in kakšne so njihove specifike. Tako uporabnikom ne bo potrebno nameščati programa Visual Studio, kar pa bo prihranilo kar nekaj časa in denarja.

Kot sem v tej nalogi že opisal, lahko delovne tokove shranjujemo v dveh različnih zapisih. Prvi je v obliki programske kode, ki poskrbi za izgradnjo aktivnosti, za povezave med njimi in njihove lastnosti. Drugi način je zapis v razširjeni XML obliki. Z uporabo urejevalnika v lastni aplikaciji lahko uporabimo le slednji način shranjevanja.



Slika 23: Gostovanje urejevalnika delovnih tokov v lastni aplikaciji.

5.2 Možnosti za uporabo WF v produktu AdLeasing

Možnosti za uporabo delovnih tokov sem iskal predvsem v modulih, ki so prilagojeni za vsako stranko posebej. Tako bi lahko obdržali splošen del programske kode enak za vse stranke, specifične dela pa bi s pomočjo delovnih tokov izločili iz splošnega dela aplikacije in s tem omogočili prilagoditve. Zaradi preprostosti načrtovanja s pomočjo grafičnega vmesnika, lahko v ta del vključimo tudi predstavnike stranke. To je tudi ena izmed zahtev pristopa BPM. Od modulov, opisanih v prejšnjem poglavju, sta se mi zdela najprimernejša modula:

- modul za odobravanje,
- modul za vodenje statusov ponudbe.

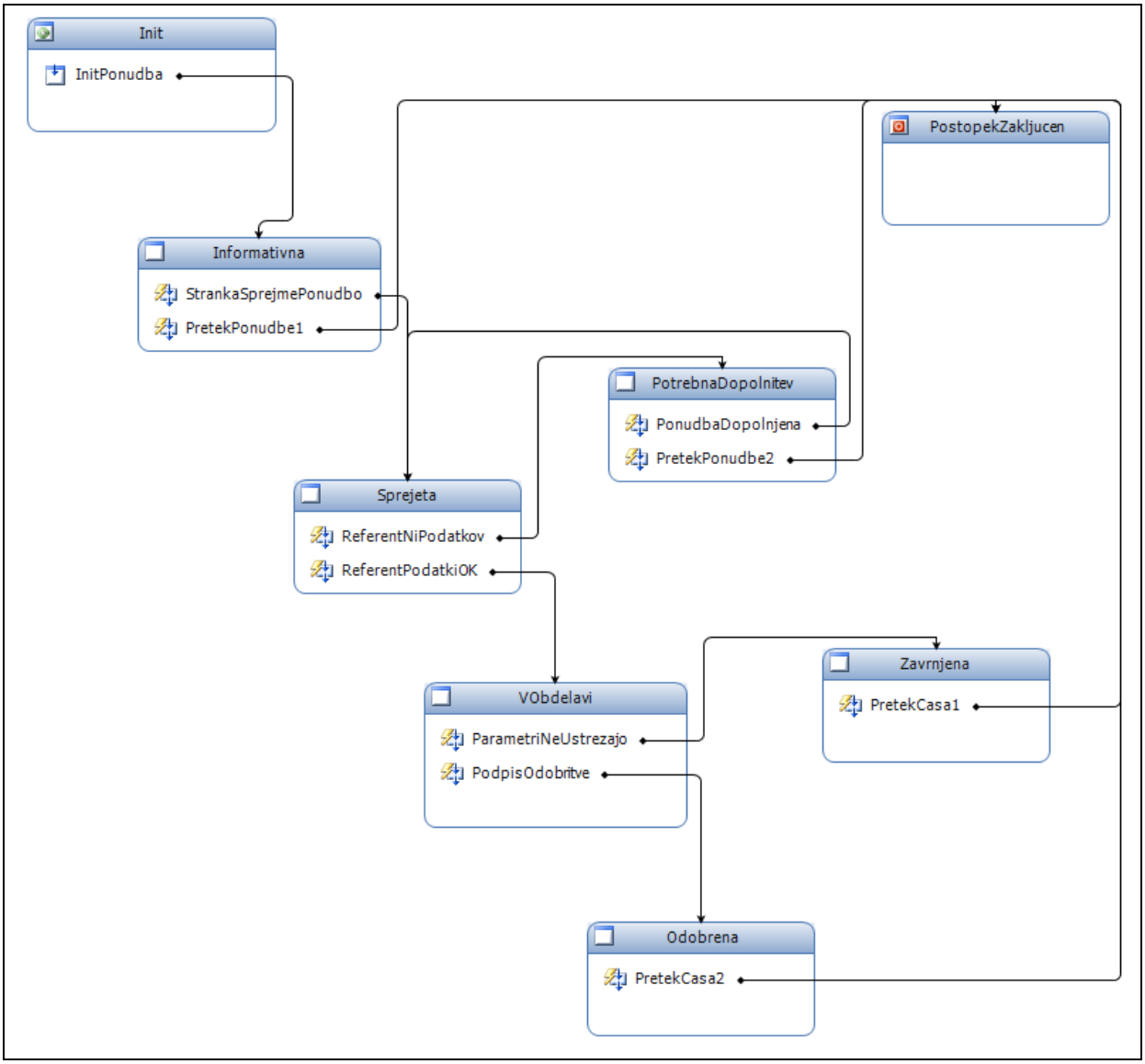
5.2.1 Uporaba v modulu za odobravanje

Kot sem že omenil v poglavju 3.4.2, je modul za odobravanje specializiran modul za vsako stranko posebej. Tako v različnih implementacijah upoštevamo povsem različne podatke, ki jih leasing hiše zajemajo za svoje stranke. Različno pa te odgovore tudi vrednotijo. Glede na to, da se računalniška ocena naredi takoj, brez posredovanja uporabnika, bomo v tem primeru uporabili sekvenčni delovni tok. Njegove značilnosti so opisane v poglavju 3.2.3.1. Za reševanje morebitnih reklamacij, bo nujno potrebna tudi uporaba sledenja, kjer bomo lahko na grafičen način videli, kako je potekal posamezni delovni tok.

Na tem mestu bi rad poudaril, da je ravno ta modul eden pomembnejših v celotni rešitvi AdLeasing. Modul za prodajo, statistiko, kontrolo in administracijo se med implementacijami ne razlikujejo prav dosti. Tako ravno modul za odobravanje in posredno za računalniško ocenjevanje stranke predstavlja poglavitni »know how« vsake leasing hiše posebej. Ravno zaradi tega ga varujejo kot poslovno skrivnost in zato o podrobnostih v tej nalogi ne morem pisati.

5.2.2 Uporaba v modulu za vodenje statusov ponudbe

V postopku prodaje leasinga ponudba potuje skozi več različnih stopenj oziroma statusov. Tudi v tem pogledu se implementacije med seboj zelo razlikujejo. Tako ima vsaka leasing hiša svoj nabor statusov ter tudi različno matriko možnih prehodov med njimi. Tukaj pa se kot posebej uporabna izkaže možnost grafične predstavitve izdelanega delovnega toka. Glede na to, da bodo v večini primerov za prehode med statusi odgovorni uporabniki in ker bo celoten proces trajal dalj časa, bomo uporabili delovne tokove tipa končni avtomat. Njihovo delovanje je opisano v poglavju 3.2.3.2. Tudi pri tem delovnem toku bo pomembno vlogo igrala funkcija sledenja poteku delovnega toka. Ker pa bodo ti tokovi od začetka do konca potekali več dni, bo nujno potrebna tudi uporaba storitve shranjevanja toka na obstojen medij. S tem bomo sproščali potreben delovni pomnilnik. Zagotovili pa bomo tudi, da se v primeru nepredvidene zaustavitve delovanja aplikacije, recimo zaradi izgube električne energije, podatki ne bodo izgubili.



Slika 24: Primer delovnega toka za vodenje statusov ponudbe

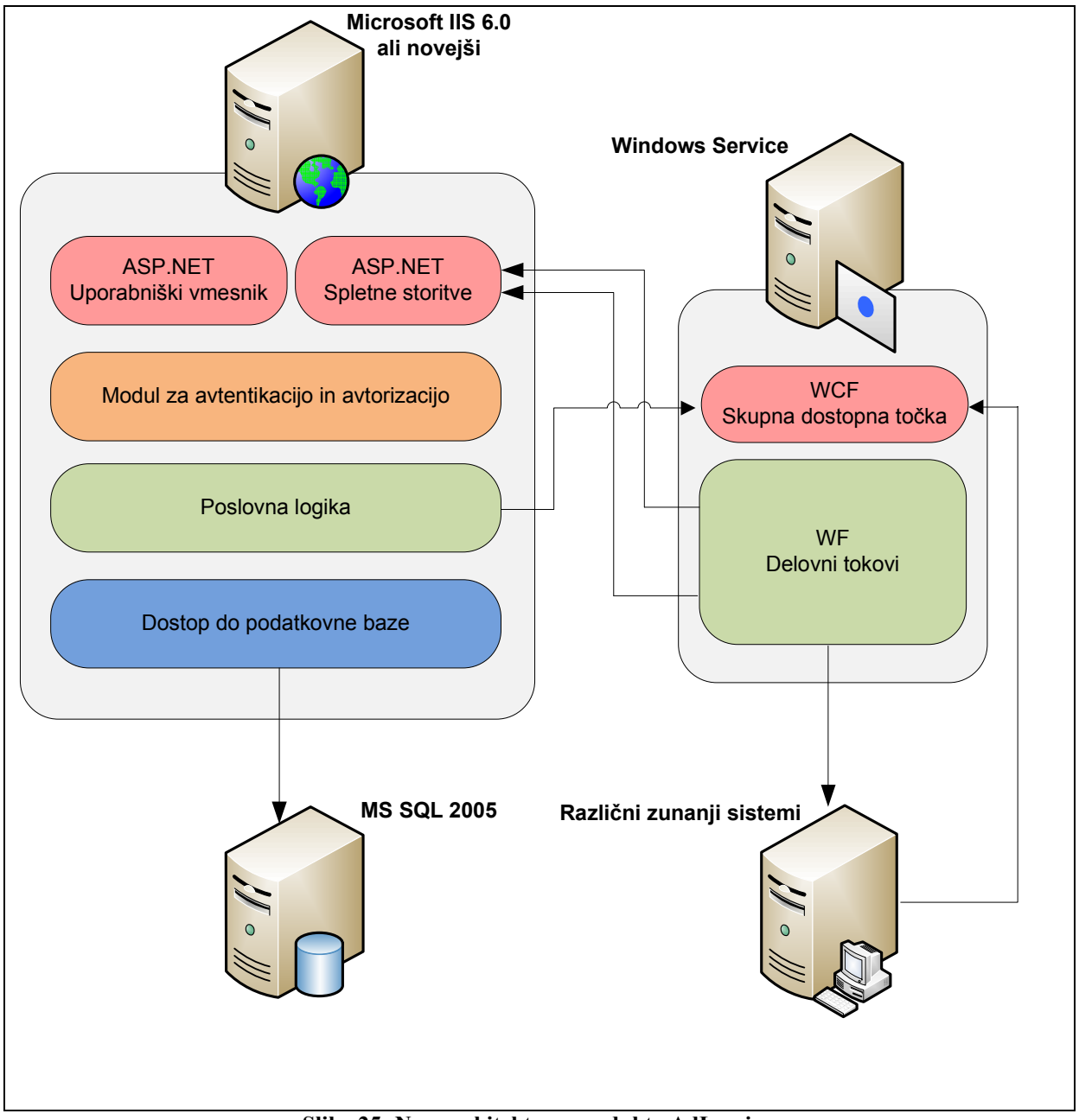
5.3 Nova arhitektura

Z uvedbo programskega modela Windows Workflow v produkt AdLeasing, se je nekoliko spremenila tudi arhitektura produkta. Tako sem iz produkta izvzel poslovno logiko, ki je bila specifična za vsako stranko posebej in jo objavil v obliki samostojnega delovnega toka. Za gostovanje le tega sem uporabil samostojen Windows Service, za komuniciranje pa WCF.

Za prenose parametrov sem uporabil črpanje podatkov preko standardiziranih podatkovnih vmesnikov. Na ta način sem lahko delovne tokove priključil tako na podatke, ki se nahajajo znotraj produkta AdLeasing, kot tudi na podatke iz zalednega sistema Globus Nova. Le ta se kot zaledni sistem pojavlja v večini slovenskih leasing hiš. Na koncu smo dodali še povezavo

na storitev Stolen Cars 24 (http://www.stolencars24.eu/), ki omogoča povpraševanje po zbirkah o ukradenih vozilih.

Poglavitna prednost takšne implementacije je samostojnost izdelanega podatkovnega toka in možnost uporabe njegovih storitev tudi v drugih aplikacijah.



Slika 25: Nova arhitektura produkta AdLeasing

6 Zaključek

Metodologija BPM postaja vsak dan pomembnejša. Temu dejstvu sledijo tudi vsi pomembnejši proizvajalci programske opreme, ki na trg pošiljajo vse več orodij za podporo BPM. V tej diplomski nalogi sem posebno pozornost namenil Microsoftovemu programskemu modelu Windows Workflow Foundation. Ugotovil sem, da le ta v celoti ne podpira dela z BPM. Vendar pa je WF sestavljeno kot ogrodje, ki omogoča dodajanje tudi lastnih aktivnosti in storitev. Zato smo na Adacti razvili nekaj dodatnih aktivnosti, in sicer aktivnost za neposreden dostop do podatkovne baze ter spremljajočih aplikacij in aplikacijo za grafičen prikaz poteka delovnega toka.

WF ni aplikacija, ampak je le ogrodje oz. nabor knjižnic, ki omogoča uporabo delovnih tokov v aplikaciji. Ker je le ta vsebovan v ogrodju .NET Framework 3.0, morajo biti tudi vse aplikacije, ki želijo WF neposredno uporabljati, narejene v omenjenem ogrodju ali novejšem. To omejitev lahko obidemo z uporabo nekaterih drugih tehnologij, kot so Windows Communication foundation, ASP.NET, itd. Bo pa takšno povezovanje prineslo nekaj dodatnega dela. Moje mnenje je, da je najprimernejša uporaba WF v povezavi z WCF. Takšen način omogoča največjo prilagodljivost in možnost uporabe polnega nabora funkcionalnosti WF.

Preden se lotimo uvajanja WF, moramo najprej vedeti, ali aplikacija, ki bo storitve WF uporabljala, ustreza kriterijem za neposredno uporabo. Ali potrebujemo hitro in odzivno uporabo WF znotraj ene aplikacije, ali pa je pomembnejša možnost za konsistentno uporabo enakih postopkov in pravil v več različnih aplikacijah.

Največja pomanjkljivost, ki jo vidim pri WF, je pomanjkanje naprednejših mehanizmov za vzdrževanje različic delovnih tokov. Tako bomo uporabljali predvsem mehanizme .NET Framework-a. Le ti pa od nas zahtevajo, da se zavedamo problema že od samega začetka razvoja delovnih tokov. Presenečen pa sem, da temu problemu literatura, ki sem jo preučil, ne namenja nobene posebne pozornosti.

7 Viri

[1] WfMC. (1999, februar). Terminology & Glossary. Številka dokumenta WFMC-TC-1011. str. 7-38. Dostopno na naslovu http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[2] (2008, januar). Business process management. Dostopno na naslovu http://en.wikipedia.org/wiki/Business_Process_Management

[3] Netcraft. (June 2008). Web Server Survey (2008 junij). Dostopno na naslovu http://news.netcraft.com/archives/2008/06/22/june_2008_web_server_survey.html

[4] Windows Workflow Foundation Mono implementation. Dostopno na naslovu http://www.mono-project.com/Workflow

[5] Andrés G Vettori. (2006, oktober). Andru's WebLog. Dostopno na naslovu http://weblogs.asp.net/andresv/archive/2006/10/20/WCF-Extensibility-_2D00_-Part-1.aspx

[6] (2008, avgust). Microsoft BizTalk Server. Dostopno na naslovu http://en.wikipedia.org/wiki/Microsoft_BizTalk_Server

[7] John Jetson, Johan Nelis. (2008). Business Process Management, Practical Guidelines to Successful Implementations. Stran 3-44.

[8] Kenn Scribner. (2007). Microsoft Windows Workflow Foundation Step by Step. Microsoft Press

[9] Bruce Bukovics. (2007). Pro WF, Windows Workflow in .NET 3.0. APress

Kazalo slik

Izjava

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorice doc. dr. Mojce Ciglarič. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljubljana, 24. septembra 2008 Jernej Kržič