UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Peter Močnik

Programski sistem za izdelavo odločb o letni odmeri dopusta

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00486/2009
Datum: 15.10.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **PETER MOČNIK**

Naslov: **PROGRAMSKI SISTEM ZA IZDELAVO ODLOČB O LETNI ODMERI DOPUSTA**

A SOFTWARE SYSTEM FOR THE CALCULATION OF EMPLOYEE'S ANNUAL LEAVE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

V okviru naloge razvijte sistem, ki za izbrano delovno organizacijo uvaja informatizirano rešitev za izdelavo odločb o letni odmeri dopusta. Pri tem skušajte rešiti problem vnosa in sprememb postopka izračuna dovoljenih dopustniških dni. V ta namen uporabite rešitev, ki bo omogočala enostavno spreminjanje postopka in usklajevanje tega s trenutno zakonodajo s stališča razvijalca, ter preglednost in razumljivost formalnega zapisa postopka s stališča uporabnika. Izdelajte pilotno implementacijo sistema in to preizkusite v delovnem okolju.

Mentor:

prof. dr. Blaž Zupan

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

tema-…

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Peter Močnik,

z vpisno številko 24950388,

sem avtor diplomskega dela z naslovom:

Programski sistem za izdelavo odločb o letni odmeri dopusta

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Blaža Zupana,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 10.12.2009 Podpis avtorja: _____

Povzetek

V podjetju Mobitel d.d. se je uporabljal zastarel sistem za izračunavanje letnih dopustov in izdajanje odločb zaposlenim. V okviru diplomske naloge smo zato razvili nov sistem, ki temelji na sodobnih tehnologijah in bistveno skrajša postopek, zagotovi brezpapirno poslovanje in večjo varnost podatkov. Sistem deluje tako, da pridobi podatke o zaposlenem iz podatkovne baze, zanj izračuna pripadajoče število dni dopusta, določi dokument v katerega zapiše rezultate in ga pošlje v sistem za elektronsko distribucijo. Zaposleni prejme elektronsko pošto z obvestilom in povezavo na dokument, ki se shrani v arhivski sistem, hkrati pa z elektronskim podpisom podpiše prejetje odločbe.

Sistem smo razvili na ogrodju Microsoft .NET z uporabo knjižnice Windows Workflow Foundation. Za distribucijo elektronskih dokumentov smo uporabili že obstoječe rešitve v podjetju. Proces računanja števila dni dopusta smo izdelali v obliki delovnih tokov, s čimer smo pridobili na preglednosti postopka. Zaradi uporabe vizualnega pristopa k sestavljanju računskih pogojev, smo dosegli boljše razumevanje postopka tako pri razvijalcih, kot pri uporabnikih sistema. Ker smo s tem olajšali pogosto spreminjanje pogojev za izračun, smo to opravilo lahko prenesli z razvojne na vzdrževalno ekipo.

Ključna prednost pred prejšnjim načinom dela je zagotovitev brezpapirnega poslovanja in s tem skrajšanje roka izdaje odločb.

Ključne besede: delovni tok, ogrodje .NET, elektronsko poslovanje, letni dopust

Abstract

The telecommunications company Mobitel Ltd employed an antiquated system for the calculation of employees' annual leave entitlements and issuing the appropriate notices. The main objective of this dissertation was to develop a new system based on modern technologies in order to substantially reduce processing time, establish a paperless operating environment and increase data security. The new system obtains the relevant employee data from a database, calculates the applicable annual leave entitlement, assigns a document into which it records the results and sends it into an electronic distribution system. The employee receives an e-mail with a notification and a link to the document stored in the archive system and can confirm its receipt with an electronic signature.

The system was developed using the Microsoft .NET Framework and the Windows Workflow Foundation library. The electronic distribution of documents was deployed through the company's existing system. The calculation of annual leave was implemented using workflows, which increased the transparency of the procedure. Using a visual approach in compiling the calculation conditions resulted in a better understanding of the procedure by both the system developers as well as the users. This approach also made frequent changes to the calculation conditions less demanding - an operation that could be relegated from the development team to the system maintenance team.

The main advantage compared to the old system is the implementation of a paperless operating environment and the accompanying reduction of time required to issue the notice documents.

Keywords: workflow, .NET framework, electronic data processing, annual leave

Zahvala

Zahvalil bi se rad
mentorju prof. dr. Blažu Zupanu,
družini za potrpežljivost,
Katji, Matjažu, Sašu, Toniju, Maji
in ostalim za vspodbudo.

Kazalo vsebine

1 Uvod

V podjetju Mobitel d.d. je nastala potreba po prenovi sistema za izdajo odločb o rednem letnem dopustu. Kadrovski sistem ni omogočal avtomatizirane izdelave odločb o dopustih, zato so odločbe delali "na roke". Ker je v podjetju več kot 1000 zaposlenih, so za to opravilo potrebovali preveč časa, hkrati pa porabili kar precej papirja.

Za rešitev problema smo si zamislili računalniško podprt sistem za izdelavo in vročanje odločb, ki je popolnoma avtomatiziran in hkrati omogoča brezpapirno poslovanje. Sistem temelji na Windows Workflow Foundation, elektronskem podpisu in e-arhivu, vsa komunikacija po poteka preko elektronske pošte. S takšno rešitvijo smo privarčevali človeške vire, kakor tudi čas potreben za vročanje odločb. Prihranili smo na papirju in hkrati zagotovili večjo preglednost vročanja in arhiviranja odločb. Zaradi pogosto spreminjajočih se pravil za odmero dopusta je bilo olajšano tudi spreminjanje pravil, saj se jih sedaj lahko spreminja "vizualno".

S programsko rešitvijo smo hoteli doseči:

- zmanjšanje števila napak,
- zmanjšanje časa potrebnega za izdelavo odločb (skrajšanje postopka),
- ukinitev papirnega poslovanja,
- ukinitev fizičnega vročanja in razvrščanja dokumentov,
- pocenitev postopka,
- prijaznejšo uporabniško rešitev,
- vpogled v arhiv e-dokumentov in
- poenostavitev spreminjanja postopka.

Pri iskanju najboljše rešitve smo izbrali pristop "vizualnega programiranja", s čemer smo dosegli preglednost postopka in poenostavili spreminjanje pravil. S takšnim pristopom smo vzdrževanje aplikacije v naslednjih letih prenesli iz razvojne v produkcijsko ekipo.

Za določene dele sistema smo uporabili že obstoječe interne rešitve, kot so sistem za elektronsko podpisovanje dokumentov imenovan Mobi podpis, Kadrovski informacijski sistem - KIS in EMC Documentum, kar nam je omogočilo hitrejšo izvedbo projekta in manj prilagajanja s strani uporabnikov. Poleg ostalega smo za rešitev problema uporabili ogrodje Microsoft .NET, Windows Workflow Foundation ter K2.NET.

2 Opis tehnologije in orodja

Za razvoj programske rešitve smo uporabili več različnih tehnologij in postopkov. Za kvalitetno razumevanje rešitve problema so v sledečih poglavjih opisani postopki in orodja, ki so bili pri delu uporabljeni.

2.1 Delovni tokovi

V splošnem je delovni tok sestavljen iz zaporedja povezanih procesnih korakov ali elementov. Na področju računalništva in informatike ga uporabljamo za modeliranje povezanosti niza operacij deklariranih kot opravila ljudi oz. skupine ljudi, organizacije osebja ali ene ali več preprostih oz. kompleksnih mehanizmov [1]. Delovne tokove lahko vidimo kot abstrakcijo dejanskih opravil v delovnem postopku. Delovni tok je način s katerim opišemo ponovljiva zaporedja operacij. Lahko rečemo tudi, da je delovni tok zaporedje aktivnosti omogočenih s sistematično izrabo virov, z definiranimi vlogami ter maso, energijo in informacijskimi tokovi združenimi v dokumentiran delovni proces. Delovni tokovi so namenjeni temu, da dosežejo namen procesa, npr. fizične transformacije, izvedbo storitev ali procesiranje informacij.

Pojem delovni tok (ang. workflow) se uporablja v računalništvu in informatiki za zajemanje in ponazoritev interakcij med ljudmi in računalniškimi sistemi. Namen sistemov za vodenje delovnih tokov je omogočanje vodenja ali opisovanja kompleksno procesiranih podatkov na vizualen način, podobno kot v diagramih poteka, vendar brez potrebe po razumevanje računalništva ali programiranja [3].

2.2 Ogrodje Microsoft.NET

V zadnjih nekaj letih sta se kot osnovi razvoja korporativnih aplikacij obdržala dva produkta. To sta Java pod okriljem podjetja Sun, in ogrodje.NET pod okriljem podjetja Microsoft. Po principu delovanja sta si precej podobna. Čeprav se sistema razvijata popolnoma ločeno, se pri obeh pojavljajo primerljive rešitve problemov, ker te pač narekuje čas in razvoj drugih tehnologij.

V podjetju Mobitel d. d. izdelujejo rešitve na obeh platformah, vendar je v Sektorju za podporo delovanja družbe poudarek na uporabi Microsoftove tehnologije, ogrodja.NET in jezika C# (ostri C oz. ang. C sharp). Ker v tem jeziku programira vsaj deset zaposlenih v

oddelku, se kot ekipa razvijalci dopolnjujejo, kar omogoča kvalitetno podporo pri delu. To dejstvo je tudi pripomoglo k odločitvi zakaj smo za platformo za izdelavo aplikacije, ki je opisana v tej diplomski nalogi, izbrali Microsoftovo okolje.

Ogrodje.NET je izdelek podjetja Microsoft in s tem operacijskega sistema MS Windows namenjeno razvoju aplikacij z objektno orientiranimi jeziki, kot sta C#, VB.NET in drugi. Ogrodje.NET je že zelo poznano v svetu razvoja programske opreme, saj obstaja že od leta 2002. Princip delovanja je podoben kot v Javi in ima prav tako konstanten razvoj. Obstajajo tako plačljiva kot brezplačna orodja za razvoj, medtem ko je sama platforma brezplačna. Ogrodje.NET ima zelo razširjeno podporo in ravno zato z njo dela ogromno število razvijalcev. Trenutno se nahaja v verziji 3.5, ki vključuje še LINQ in ADO.NET Entity Framework ter se razvija še naprej. Sestavljeno je iz dveh osnovnih komponent. To sta Common Language Runtime (CLR) in .NET Framework class library.

CLR je vmesno okolje v katerem se programi napisani v npr. C# izvajajo. .NET Framework class library pa je knjižnica metod s katerimi pišemo programsko opremo. CLR skrbi za izvajanje izvorne kode, za komunikacijo s pomnilnikom, bdi nad delovanjem niti in komunicira z nižjimi nivoji operacijskega sistema [5].

Knjižnica razredov je zbirka metod in tipov, ki jih potrebujemo za razvoj aplikacij. To velja tako za razvoj konzolnih aplikacij kot tudi za razvoj aplikacij z grafičnim vmesnikom ali razvoj spletnih aplikacij. Knjižnico uporabljajo orodja Windows Presentation Foundation, ki je zbirka metod za delo z grafiko, Windows Communication Foundation za komunikacijo med procesi in Workflow Foundation za opisovanje delovnih tokov. Slednje orodje smo uporabili pri izdelavi v pričujočem delu predstavljene aplikacije.

Vsa izvorna koda napisana v kateremkoli .NET jeziku se pred izvajanjem prevede v CIL (common intermediate language), ki pa se ne izvede dokler ni dejansko potrebno. Šele ob izvedbi se sproži t.i. JIT (just-in-time compiler), ki CIL kodo prevede v izvršno kodo sistema.

Uporabniki ki uporabljajo spletne aplikacije razvite na ogrodju .NET ne zaznajo nobene razlike v primerjavi z drugimi tehnologijami, ki se uporabljajo za razvoj spletnih aplikacij, saj se stran narejena v okolju ASP.NET obnaša kot vsaka druga spletna stran. Uporabniki aplikacij za okna morajo pred uporabo namestiti še knjižnice .NET, ki so brezplačno na voljo na svetovnem spletu.

2.3 Windows Workflow Foundation ali Osnova za delovne tokove

Da lahko pojasnimo, kako je bila izdelana rešitev oz. programski sistem, si moramo podrobneje ogledati osnovo na kateri deluje. V tem podpoglavju bomo opisali tehnologijo za izdelavo delovnih tokov in principe na katerih delujejo.

2.3.1 Osnove Windows Workflow Foundation

Windows Workflow Foundation (v nadaljevanju WWF) je del ogrodja.NET, ki je trenutno v verziji 3.0 in je bilo izdano v letu 2006. Namenjeno je izdelavi aplikacij, ki bazirajo na delovnih tokovih. Vsebuje zbirko orodij za oblikovanje in uporabo delovnih tokov, programski model za nadzor in komunikacijo z delovnimi tokovi, center za pravila, izvajalno okolje (ang. runtime engine) za izvajanje delovnih tokov in zbirko storitev za spremljanje izvajanja, upravljanje s transakcijami ipd.

Sistemi delovnih tokov se uporabljajo za avtomatizacijo poslovnih procesov zaradi povečanja natančnosti, učinkovitosti in predvidljivosti rezultatov logičnih aktivnosti znotraj procesa. Delovni tok poveča preglednost zapletenih procesov in omogoča boljše sodelovanje med poslovnimi uporabniki in tehnologijo [3].

WWF je razvojno orodje, ki poenostavlja avtomatizacijo poslovnih procesov za okolje Windows. Tako omogoča:

- sekvenčne diagrame ali avtomate,
- uporabo načrtovalnika delovnih tokov znotraj okolja Visual Studio,
- vizualizacijo delovnih tokov za lažjo predstavo poslovnim uporabnikom,
- spremljanje in urejanje stanja izvajanja delovnih tokov z vgrajenimi metodami in
- vgradnjo človeških delovnih tokov v različnih fazah izvajanja.

Prednosti razvoja aplikacij s tehnologijo delovnih tokov so:

- možnost odločanja na osnovi poslovnih pravil. Ta so lahko preprosta, kot odločitve tipa da/ne, in kompleksnejša kot npr. pogojni stavki,
- možnost komunikacije z drugimi aplikacijami zunaj samega delovnega toka, kot tudi interakcija s človeškimi viri,
- vzdrževanje stanja skozi življenjski cikel delovnega toka. Če npr. neka aktivnost traja predolgo, lahko ugotovimo kje je prišlo do zaostanka,

- shranjevanje stanja delovnega toka in kasneje nadaljevanje od te točke naprej,
- komponentni pristop k razvoju programske opreme. Tako je lahko vsak korak izveden kot del programske opreme. Možno je izdelati v naprej definirane korake, ki so uporabni v celotni aplikaciji in jih uporabiti v različnih delovnih tokovih,
- izdelava in spreminjanje delovnih tokov v grafični obliki, tako da so bolj preprosto prikazani posamezni elementi in relacije med njimi,
- možnost spremljanja izvajanja delovnega toka v realnem času,
- možnost spreminjanja delovnega toka v času izvedbe.

2.3.2 Kaj omogoča Windows Workflow Foundation?

Najpomembnejše lastnosti Windows Workflow Foundation so:

- skupna platforma za delo z delovnimi tokovi v okolju Windows,
- platforma za raznolike aplikacije z delovnimi tokovi,
- združevanje sistemskih in človeških delovnih tokov.

2.3.3 Skupna platforma za delovne tokove v okolju Windows

Delovni tokovi so bili že pred Windows Workflow Foundation prisotni v mnogih aplikacijah, vendar do sedaj za njih ni bilo enotne podpore. Zato so pri Microsoftu spremenili koncept tako, da bo od zdaj naprej v uporabi le še en sistem za delovne tokove, ki je že zdaj integralni del okolja Windows. V aplikacijah, ki temeljijo na ogrodju .NET od verzije 3.0 naprej, je ta tehnologija na voljo za uporabo, kar je prikazano na sliki 1.

Slika 1: Oris tehnologije Windows Workflow Foundation

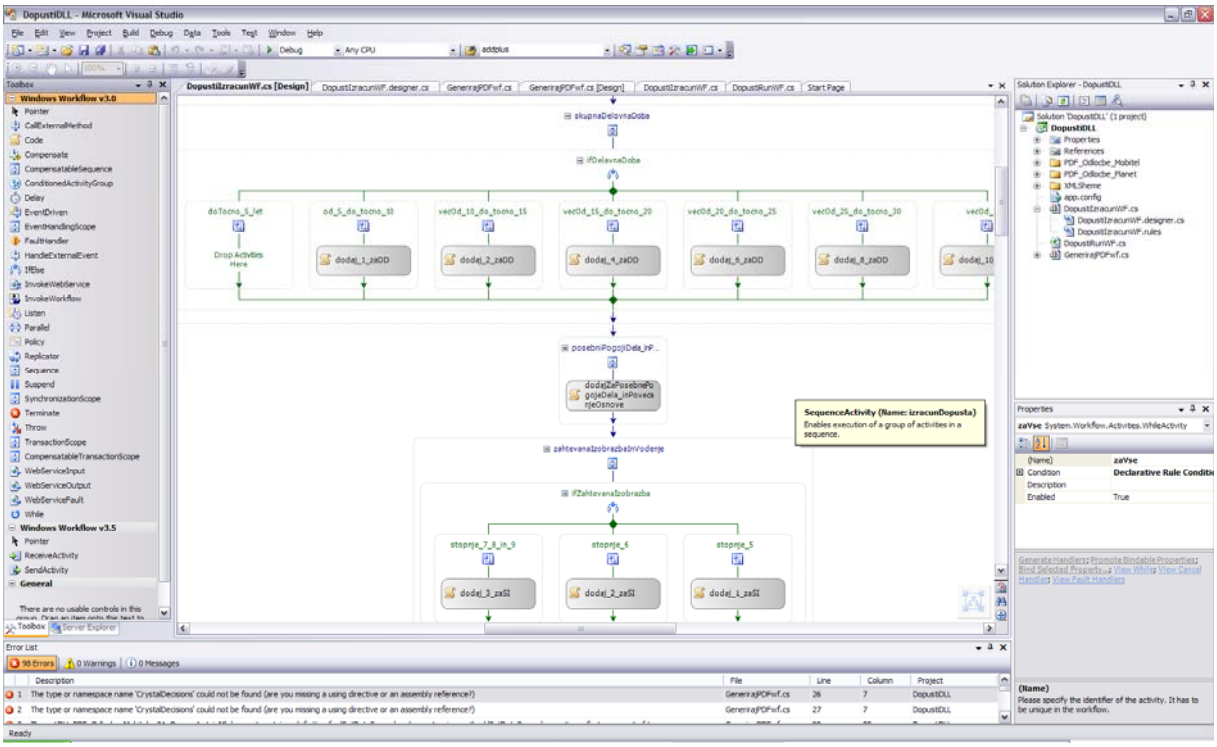
Windows Workflow Foundation je primarno namenjen razvijalcem in ne poslovnim uporabnikom. Za delo z Windows Workflow Foundation je potrebno znanje programerja, saj tehnologija ne vključuje orodij za spremljanje in spreminjanje delovnih tokov, temveč je v ozadju vedno potrebno kodiranje. Razvijalci lahko po potrebi sami razvijejo potrebne funkcije za poslovne uporabnike, vendar je okolje primarno namenjeno le razvijalcem.

2.3.4 Platforma za raznolike aplikacije z delovnimi tokovi

Delovni tok je sestavljen iz skupin aktivnosti, kjer vsaka aktivnost pomeni izvajanje neke akcije. Windows Workflow Foundation je zbirka splošnih aktivnosti za izvajanje delovnega toka. Osnovna knjižnica aktivnosti vsebuje elemente, kot so če/potem (ang. if/else) in dokler (ang. while) zanke, kot tudi izvajalno okolje, podporo za komunikacijo z drugimi aplikacijami (WCF – Windows Communication Foundation) in ostalo.

Čeprav vgrajene aktivnosti omogočajo precej, lahko programerji implementirajo svoje aktivnosti odvisno od potreb.

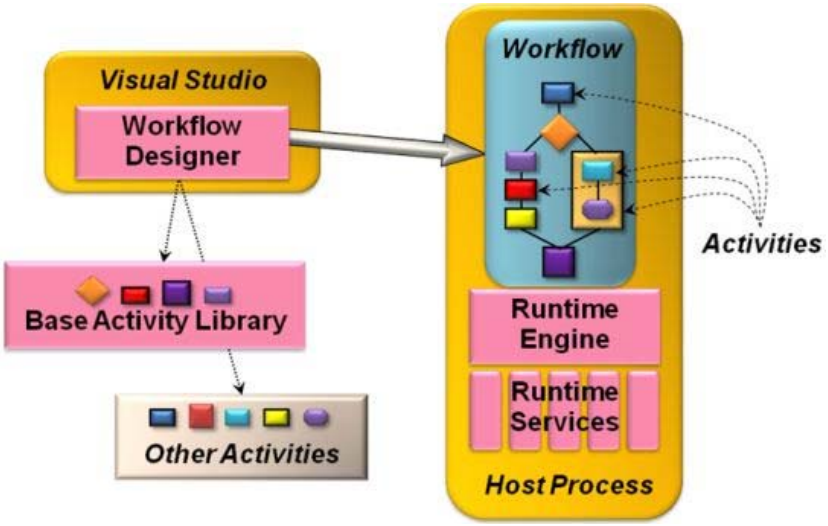
Delovni tokovi so lahko napisani neposredno v izvorni kodi. Bolj praktično je grafično oblikovanje in dodajanje izvorne kode le po potrebi. Zato okolje Visual Studio vsebuje orodje Workflow Designer (slika 2) za načrtovanje in razvoj delovnih tokov. To orodje vsebuje velik nabor vizualnih komponent za delo s tokovi, ki se jih lahko uporablja na način povleci in spusti. Vsaka od komponent ima svoje lastnosti, ki so programabilne.



Slika 2: Orodje za delo z delovnimi tokovi

Osnovne komponente, ki jih podpira Windows Workflow Foundation, so (slika 3):

- aktivnost (ang. Activity): delovna enota. Lahko je enostavna ali kompleksna,
- delovni tok (ang. Workflow): skupina aktivnosti, ki vsebujejo neko procesno logiko,
- orodje za izdelavo tokov (ang. Workflow designer): grafično orodje, tipično del Visual Studia, ki se uporablja za izdelavo delovnih tokov,
- knjižnica osnovnih aktivnosti (ang. Base activity library): skupina osnovnih aktivnosti, poleg katerih se lahko uporabljajo tudi druge aktivnosti,
- izvajalno okolje (ang. Runtime Engine): knjižnica funkcij za izvajanje delovnih tokov, ki skrbi tudi za komunikacijo z drugimi procesi,
- izvajalne storitve (ang. Runtime services): skupina storitev, ki jih lahko delovni tokovi uporabljajo s podporo za transakcije, hranjenje stanj in spremljanje izvajanja,
- gostiteljski proces (ang. Host process): aplikacija okolja Windows v okviru katere teče izvajalni stroj delovnih tokov in tokovi, ki jih izvaja.



Slika 3: Osnove komponente platforme WWF

2.3.5 Združevanje sistemskih in človeških delovnih tokov

Čeprav poslovni procesi pogosto vključujejo ljudi in aplikacije, je avtomatizacija interakcij med aplikacijami nekaj popolnoma drugega, kot avtomatizacija interakcij med ljudmi. Zato obstajata dva načina za izdelavo delovnih tokov. Interakcije med aplikacijami so po navadi predvidljive in relativno statične. Logika, ki usmerja te interakcije, je lahko definirana enkrat in potem brez sprememb večkrat uporabljena. Sistemski delovni tokovi so po navadi definirani za izmenjavo jasno definiranih podatkov, lepo definiranih podatkovnih struktur, kot so npr. dokumenti XML, ki jih je enostavno računalniško obdelati brez posegov ljudi.

V nasprotju s slednjimi, človeški delovni tokovi koordinirajo interakcije med ljudmi. Aplikacije, ki komunicirajo z ljudmi ali namesto njih, morajo biti mnogo bolj fleksibilne, kot tiste, ki komunicirajo izključno s programsko opremo. Ljudje si pogosto premislijo, dodajajo nove poti in izjeme, odločijo se nenadno in nepredvideno prekinejo postopek in počnejo stvari, ki naredijo efektivne človeške delovne tokove veliko bolj dinamične, kot so sistemski delovni tokovi. Zato naj bi bile aplikacije, ki podpirajo človeške delovne tokove bolj dogodkovno naravnane in naj bi imele bolj fleksibilen pristop. Človeški delovni tokovi po navadi vsebujejo slabše strukturirane podatke, ki jih lahko razumemo ljudje, to so elektronska pošta, dokumenti z besedilom ipd., za razliko od dobro strukturiranih informacij, ki jih uporabljajo sistemski delovni tokovi.

Tehnologija Windows Workflow Foundation je osredotočena na reševanje obeh tipov delovnih tokov na enoten način. Tako vsebuje sekvenčni in dogodkovno orientirani pristop k definiranju delovnih tokov z možnostjo dodajanja korakov in spreminjanja delovnih tokov.

2.3.6 Razumevanje delovnih tokov

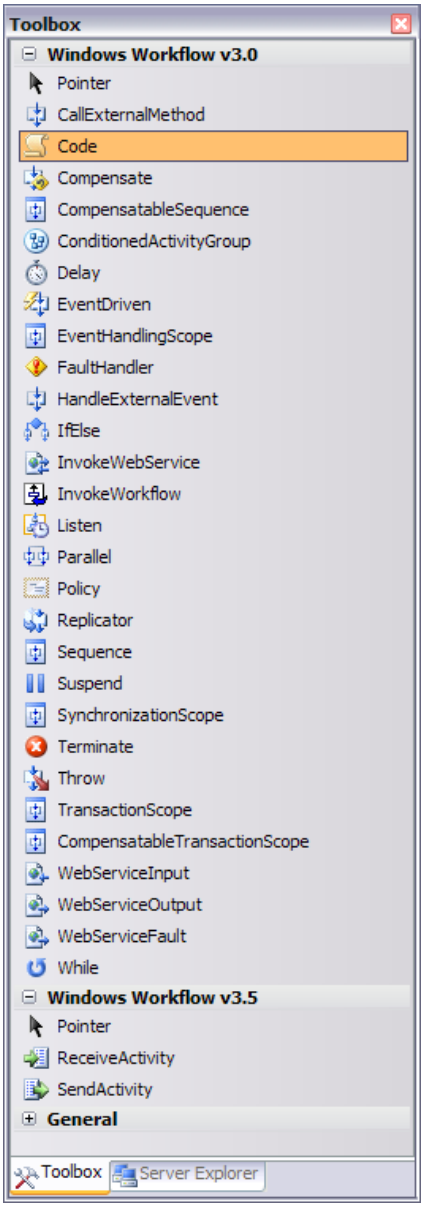
Vsak delovni tok je sestavljen iz aktivnosti, ki opravljajo neko funkcijo znotraj tega toka. Delovni tok je torej množica aktivnosti s svojim življenjskim ciklom in vrstnim redom izvajanja. Sistemski delovni tokovi izvajajo aktivnosti na jasno definiran in predvidljiv način, medtem ko jih človeški delovni tokovi ne. Za rešitev obeh problemov lahko uporabimo sekvenčne delovne tokove, ki izvajajo aktivnosti po prej definiranem vzoru, in končne avtomate, ki se odzivajo na zunanje dogodke, ko se ti pojavijo. Oboji tečejo v istem okolju in oboji lahko uporabljajo iste prirejene aktivnosti. Sekvenčni pristop je bolj primeren za sistemske delovne tokove, medtem ko so končni avtomati bolj primerni za bolj rahlo definirane človeške delovne tokove.

2.3.7 Sekvenčni delovni tokovi

Sekvenčni delovni tokovi so namenjeni uporabi v aplikacijah, kjer se aktivnosti izvajajo v jasno definiranem vrstnem redu. Vrstni red lahko vsebuje zanke, razvejitve in druge kontrole toka, v vsakem primeru pa je zaporedje izvajanja znano od začetka do konca. Osnovna knjižnica aktivnosti v Windows Workflow Foundation vsebuje elemente (prikazane tudi na sliki 4):

- če/potem (ang. if/else): izvede aktivnost po dveh ali več možnih poteh, glede na izpolnjen pogoj,
- medtem (ang. while): ponavlja aktivnost, dokler je pogoj izpolnjen,
- zaporedje (ang. sequence): enkrat izvede množico aktivnosti v določenem vrstnem redu,
- vzporedno (ang. paralell): izvede dve ali več aktivnosti naenkrat in počaka, da se vse končajo,
- koda (ang. code): izvede del izvorne kode,
- nadomestitvena logika (ang. Compensation Handler): vsebuje nadomestitveno logiko, recimo nekaj kar se izvede ob napaki,
- poslušaj (ang. listen): čaka na določen dogodek, potem izvede aktivnost,
- zadrži (ang. delay): zadrži izvajanje delovnega toka za določen čas,
- klic zunanje metode (ang. CallExternalMethod): pokliče metodo objekta v aplikaciji, ki ni del delovnega toka,

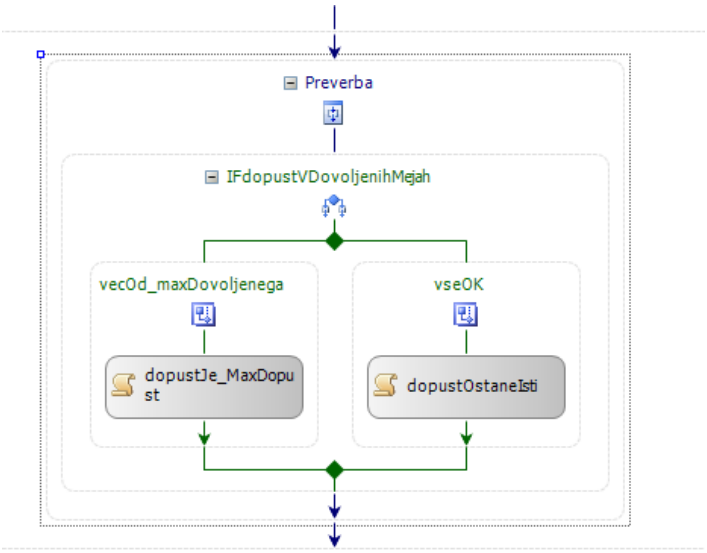
- obravnavaj zunanji dogodek (ang. HandleExternalEvent): čaka na klic iz druge metode znotraj aplikacije ampak izven delovnega toka,
- začetni delovni tok (ang. InvokeWorkflow): zažene nek drug delovni tok,
- zaženi spletno storitev (ang. InvokeWebService): pokliče spletno storitev,
- združevanje transakcij (ang. TransactionScope): omogoča združevanje več aktivnosti v eno samo,
- prekini (ang. Terminate): preneha z izvajanjem delovnega toka.



Slika 4: Orodjarna Visual Studia 2008 z naborom elementov za izdelavo delovnih tokov

Osnovne aktivnosti Windows Workflow Foundation so podobne tistim v BPEL-u (Business Process Execution Language), ki sta ga na začetku zasnovala IBM in Microsoft, zdaj pa so del

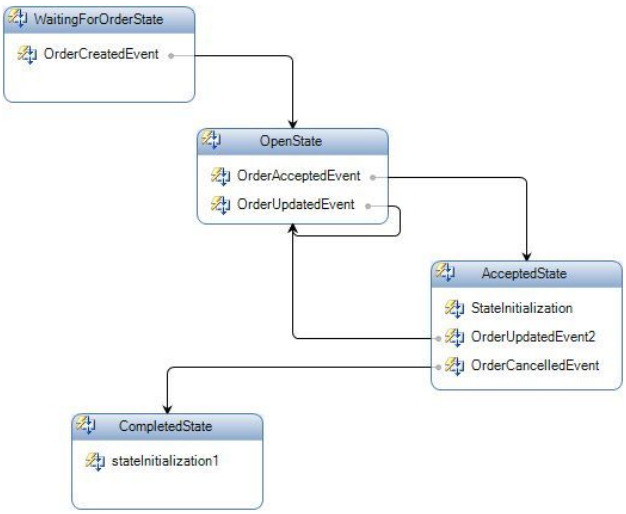
standarda OASIS. Delovni tokovi Windows Workflow Foundation so z BPEL-om kompatibilni. Primer razvejitve v delovnem toku je prikazan na sliki 5.



Slika 5: Primer sekvenčnega delovnega toka narejenega v orodju za izdelavo tokov

2.3.8 Končni avtomat

Za razliko od sekvenčnega delovnega toka, pri katerem imajo aktivnosti določeno zaporedje, ima končni avtomat aktivnosti v obliki avtomata končnih stanj. Ta deluje tako, da ima definirana stanja in dogodke, ki sprožajo prehode med temi stanji (slika 6).



Slika 6: primer končnega avtomata

Uporaba takšnega delovnega toka je primerna, kadar natančno zaporedje dogodkov ni znano v naprej, npr. pri procesih, ki jih bolj določajo nepredvidljivi dogodki, kot pa zaporedje dogodkov ali kadar je število možnih poti preveliko za praktično rešitev. Najpogosteje pride do takšnih primerov, kadar so v proces kot uporabniki vpleteni ljudje, ne pa računalniški sistemi. Preskakovanje dogodkov je preprosto, prav tako tudi zaključevanje procesa v katerem koli trenutku. Končni avtomati nam olajšajo opis delovnega procesa glede na potrebe in pravila neke organizacije. V takšnih primerih nam koristijo pri boljšem sodelovanju in razumevanju med oblikovalci procesa in razvijalci programske rešitve.

Osnovna knjižnica vsebuje posebej definirane aktivnosti za končni avtomat. Te aktivnosti so:

- stanje (ang. State): predstavlja stanje v končnem avtomatu delovnega toka,
- dogodkovno voden (ang. EventDriven): definira prehod z eno ali več aktivnostmi, ki naj se zgodijo, ko pride do določenega dogodka in je avtomat v določenem stanju,
- nastavi stanje (ang. SetState): spremeni stanje avtomata delovnega toka,
- inicializacija stanja (ang. StateInitialization): definira eno ali več aktivnosti, ki naj se zgodijo same po sebi, ko pride avtomat v določeno stanje,
- finalizacija stanja (ang. StateFinalization): definira eno ali več aktivnosti, ki naj se zgodijo same po sebi, ko gre avtomat iz določenega stanja.

Stanja lahko tudi gnezdimo. Dogodek definiran v zunanjem stanju velja tako za to stanje, kakor tudi za vsa stanja, ki jih vsebuje. Gnezdenje omogoča enostaven način dodajanja dogodka, ki se lahko zgodi kadarkoli oz. v katerem koli stanju. Ker lahko avtomati vsebujejo tudi sekvence, se lahko ob vsakem prehodu stanja izvede neko zaporedje aktivnosti.

Z združevanjem teh dveh tipov delovnih tokov lahko kvalitetno pokrijemo različne potrebe tako sistemskih, kakor tudi človeških delovnih tokov.

2.3.9 Izdelava delovnih tokov

Delovni tok najlaže naredimo v orodju za izdelavo tokov. Z načinom povleci in spusti postavimo aktivnosti na podlago, jih med sabo povežemo in jim določimo lastnosti. Nato napišemo izvorno kodo, ki obravnava vsako aktivnost.

Seveda lahko delovni tok napišemo samo s pomočjo izvorne kode, kar na koncu tudi je. Windows Workflow Foundation je samo razred in trije imenski prostori znotraj tega razreda nam omogočijo vso funkcionalnost.

Imenski prostori so:

- System.Workflow.Activities,
- System.Workflow.ComponentModel in
- System.Workflow.Runtime.

Primer osnovne izvorne kode:

```
using System.Workflow.Activities;
public class ExampleWorkflow : SequentialWorkflow
{
 …
}
```

Ko se izvede konstruktor, se instancirajo konfigurirane aktivnosti tega delovnega toka.

Delovni tokovi se lahko zapišejo tudi v obliki XAML, kar je okrajšava za eXtensible Application Markup Language.

Primer zapisa v obliki XAML:

```
<?Mapping XmlNamespace="Activities"
 ClrNamespace="System.Workflow.Activities"
 Assembly="System.Workflow.Activities" ?>
<SequentialWorkflow x:Class="ExampleWorkflow"
 xmlns="Activities" xmlns:x="Definition">
 …
</SequentialWorkflow>
```

2.3.10 Izdelava aktivnosti

Aktivnosti so osnovne sestavine delovnih tokov. Lahko so zelo preproste, kot je npr. začetek ali konec, ali pa bistveno kompleksnejše, kot je npr. če/potem (ang. if/else) ali zanke. Aktivnost je lahko narejena na novo ali pa je razširjena oblika obstoječe. Prav tako lahko več aktivnosti združimo v eno samo. Navsezadnje je celoten delovni tok prav tako aktivnost.

Za izdelavo aktivnosti lahko uporabimo vizualno orodje (Activity Designer tool) ali pa napišemo izvorno kodo.

Primer izvorne kode za opis aktivnosti:

```
using System.Workflow.ComponetModel;
public class ExampleActivity : Activity
{
 …
}
```

2.3.11 Uporaba pogojev in pravil

V delovnem toku je vsak če/potem stavek ali vsaka zanka pravzaprav poslovno pravilo. Ta pravila lahko zapišemo na dva različna načina. Pogoj lahko zapišemo neposredno v izvorni kodi, kar se imenuje "code condition". Napišemo metodo, ki glede na dane pogoje vrne binarni rezultat. Namesto tega lahko uporabimo drug pristop tako, da definiramo "rule condition". Za definicijo lahko uporabimo orodje (Rule Condition Editor) ali pa ga definiramo neposredno v kodi. Pogoji pravil so shranjeni v obliki XML v posebni datoteki in se evalvirajo ob izvajanju izvorne kode. To pomeni, da jih lahko spreminjamo medtem, ko aplikacija teče. Isto pravilo lahko potem uporabimo v več delovnih tokovih.

2.3.12 Združevanje pogojev in aktivnosti

Aktivnosti, ki bazirajo na pogojih in zankah, predstavljajo enostaven način izvajanja delovnih tokov. Za bolj zapletene situacije, ki vključujejo tudi interakcijo z ljudmi, pa je bolje, da uporabimo skupine pogojenih aktivnosti (ang. Conditioned Activity Groups). Te skupine vsebujejo aktivnosti, kjer ima vsak ko (ang. when) izrecen pogoj. Ob izvajanju se vsi pogoji preverijo. Za vsak izpolnjen pogoj se izvedejo pripadajoče aktivnosti, ki so lahko poljubno kompleksne. Evalvirajo se le enkrat in nato čakajo, da so vsi pogoji izpolnjeni. V bistvu so te skupine pomanjšane verzije samostojnih delovnih tokov in v našem primeru niso toliko aktualne.

2.3.13 Izvedbeno okolje (ang. Runtime engine)

Vsako izvajanje delovnega toka se zgodi v izvedbenem okolju (ang. Runtime Engine). Ta izvaja delovne tokove in upravlja z njihovimi stanji v njihovem življenjskem ciklu. Izvedbeno okolje je knjižnica, ki teče v Windows okolju, ne glede na to ali gre za konzolno aplikacijo ali spletno storitev. Edini pogoj za delovanje je v delovni pomnilnik naloženo ogrodje .NET 3.0.

Vseeno pa je treba biti pozoren na nekatere posebnosti delovnih tokov. Delovni tokovi lahko tečejo zelo dolgo časa. Izvedbeno okolje mora biti zato sposobno shraniti parametre, kar pomeni, da se lahko delovni tok zažene od prekinitve naprej in s tem privarčuje z viri. Delovni tok poženemo z izvorno kodo.

Primer izvorne kode za zagon delovnega toka:

```
using System.Workflow.Runtime;
class ExampleHost
{
 static void Main()
 {
 WorkflowRuntime runtime = new WorkflowRuntime();
 runtime.StartRuntime();
 runtime.StartWorkflow(typeof(ExampleWorkflow));
 …
 }
}
```

2.3.14 Spremljanje izvajanja

Ker so delovni tokovi zgrajeni iz aktivnosti, je relativno enostavno spremljati njihovo izvajanje. Windows workflow foundation omogoča spremljanje statistike izvajanja aktivnosti. Katere vrednosti naj se spremljajo definiramo ob razvoju. Izvedbeni stroj podatke shranjuje v SQL podatkovno bazo, kjer so nam nato na voljo za uporabo. Pomanjkljivost Windows workflow foundation je v tem, da ne premore orodij ali metod za analize, vendar jih po potrebi lahko napišemo sami.

2.4 MS SQL Strežnik 2005

MS SQL Strežnik 2005 (ang. MS SQL Server 2005) je relacijska podatkovna baza podjetja Microsoft s poizvedbenima jezikoma ANSI SQL in T-SQL. V našem podjetju trenutno uporabljamo različico 2005, ki je postavljena v gručo, zaradi boljšega izkoristka strojne opreme. Obstaja že različica 2008, ki ne prinaša bistvenih prednosti.

MS SQL Strežnik 2005 podpira vse standardne SQL operacije, poleg tega pa nudi še celovito podporo za podatkovne transakcije v obliki XML. To je v rešitvi našega problema bistvena prednost, saj smo s pomočjo komunikacije v obliki XML občutno skrajšali čas razvoja. Podatke, ki so v podatkovni bazi sicer shranjeni v obliki tabele, izvozimo v obliki XML.

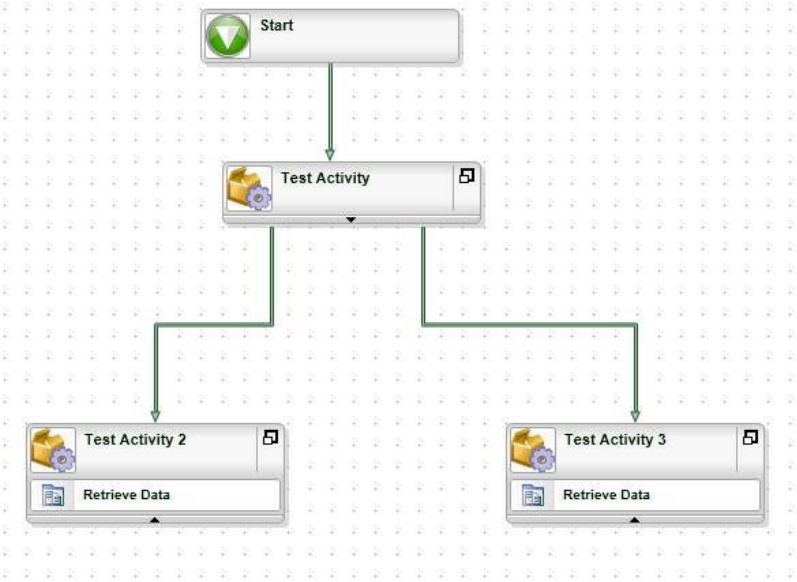
Datoteko oblike XML nato uporabimo neposredno v našem programu, ki je s tem neodvisen od konkretne podatkovne baze.

V podjetju na strežniku MS SQL 2005 teče tudi večina drugih aplikacij. Ker za strežnik skrbi posebna skupina zaposlenih, ki zanj nudi vzdrževanje in podporo, skrbi za nadgradnje in izdeluje varnostne kopije, je bil to najbolj smiselni izbor.

2.5 Sistem K2.NET BlackPearl

K2.NET BlackPearl je obširen sistem za gradnjo procesno vodenih aplikacij v korporativnih okoljih. V prvi vrsti gre za samostojno business process management (BPM) okolje. Za razliko od Windows Workflow Foundation, ki je bolj razvojno orodje, omogoča razvojno integracijo z Visual Studiem in SharePoint-om, ki ni omejena samo na razvijalce, ampak omogoča tudi sodelovanje uporabnikov.

V podjetju se uporablja za vodenje postopkov ali procesov, ki vključujejo veliko število ljudi, in za izredno kompleksne postopke, kot je nabava strojne opreme, programske opreme ali nabava drobnega inventarja (slika 7). Povezuje se tako s kadrovskim sistemom in dokumentnim strežnikom, kakor tudi z Active Directory Services in ostalimi sistemi.



Slika 7: Primer podatkovnega toka v K2 BlackPearl

Tudi K2.NET BlackPearl bazira na ogrodju .NET. Jedro njegovega delovanje je prav tako Windows Workflow Foundation. Ker pa je K2.NET sistem zgrajen na strežniški arhitekturi, potrebuje lasten strežnik, kar za srednje veliko podjetje nikakor ni poceni. V vsakem primeru je sistem preveč kompleksen za izdelavo delovnih tokov, ki jih potrebujemo v aplikaciji za dopuste. Ker se sistem v podjetju že uporablja za nabavo, je izredno praktičen za dostavo izdelanih odločb in podpisovanje dokumentov.

2.6 Digitalno podpisovanje dokumentov

Za podpisovanje dokumentov je bila v podjetju izdelana aplikacija Mobi podpis**.** Temelji na ogrodju .NET in rešitvi ClickOnce. Iz vidika aplikacije za dopuste je pomembno to, da zaposleni ob zahtevi za podpis dobi elektronsko pošto s povezavo na Mobi podpis. Ob kliku na povezavo se požene aplikacija, v kateri uporabnika pričaka seznam dokumentov, ki jih mora podpisati. Uporabnik lahko dokumente pregleda in vsakega posebej podpiše. Omogočeno je tudi podpisovanje cele skupine dokumentov, kot so npr. odločbe o odmeri dopusta za celo podjetje. Bistveno je to, da se z enim klikom lahko podpiše vse dokumente naenkrat, kar prihrani ogromno dela in časa.

Podjetje je vsakemu zaposlenemu pridobilo digitalni certifikat, ki se uporablja tudi za druge namene in ne samo za podpisovanje dokumentov. Digitalni podpis deluje na principu javnega in privatnega ključa. Vsak zaposleni v podjetju ima na svojem računalniku shranjen digitalni certifikat s privatnim ključem. Kadar ima zaposleni svoj osebni certifikat, ki ga je pridobil pri drugih overjenih izdajateljih, kot so npr. SIGEN-CA, POSTAR-CA, NLB-CA ali drugih, lahko dokument podpiše s katerimkoli od teh ključev, dokler so veljavni.

Aplikacija MobiPodpis je povezana z delovnim tokom K2.NET, tako se lahko iz procesa za izdajo dopustov kadarkoli pokliče zahtevo za digitalni podpis.

Digitalni podpis se shrani v obliki Cryptographic Message Syntax v dokument v obliki PDF. Podpis je lahko tudi vizualno prikazan na strani, zatem ko je bila nad dokumentom izvedena zgostitvena funkcija SH1. Na ta način se po potrebi ugotovi ali je bil dokument naknadno spremenjen. Poleg tega dobi podpis še časovni žig, da se ve, kdaj je bil podpisan.

2.7 Dokumentni sistem EMC Documentum

EMC Documentum je korporativni sistem za upravljanje z vsebinami. Certificiran je za upravljanje z dokumenti in je narejen v skladu s standardom ISO 9001. V podjetju je nameščen skupaj z diskovnim poljem Centera, ki je prav tako produkt podjetja EMC, kamor se shranjujejo dokumenti.

Po zakonu o elelektronskem poslovanju in elektronskem podpisu (ZEPEP) lahko nadomestimo tiskan papir z ustrezno elektronsko obliko, če so izpolnjeni pogoji, opisani v 12. členu tega zakona [7]. Ti pogoji so:

- da so podatki, vsebovani v elektronskem dokumentu ali zapisu, dosegljivi in primerni za kasnejšo uporabo,
- da so podatki shranjeni v obliki, v kateri so bili oblikovani, poslani ali prejeti, ali v kakšni drugi obliki, ki verodostojno predstavlja oblikovane, poslane ali prejete podatke,
- da je iz shranjenega elektronskega sporočila mogoče ugotoviti od kod izvira, komu je bilo poslano ter čas in kraj njegovega pošiljanja ali prejema in
- da uporabljena tehnologija in postopki v zadostni meri onemogočajo spremembo ali izbris podatkov, ki ju ne bi bilo mogoče enostavno ugotoviti, oziroma obstaja zanesljivo jamstvo glede nespremenljivosti sporočila.

Čeprav EMC Documentum v širši konfiguraciji omogoča tudi postavljanje delovnih tokov, zajem dokumentov in še mnogo več, v našem podjetju uporabljamo le sistem za arhiviranje in zaščito dokumentov. Elektronski arhiv, ki ga uporabljamo, je v procesu certifikacije Arhiva Slovenije. S tem bi radi dosegli popolno ukinitev papirnega poslovanja, kar je dolgoročni cilj podjetja, in vse dokumente shranjevali v ta sistem v digitalni obliki.

Za aplikacijo za dopuste je pomembno to, da so dokumenti, ki so shranjeni v sistem, vedno dostopni avtorju, lastniku ali administratorju. Uporabnik ima dostop do dokumentov glede na pravice, ki jih ima oz. potrebuje. Zato so dostopi zaščiteni s sistemskim geslom. Sami dokumenti imajo življenjsko dobo, kar pomeni, da jih sistem po pretečenem roku trajanja sam reciklira oz. izbriše. Pri odločbah o odmeri letnega dopusta je določen rok hranjenja dokumentov tekoče leto in še dodatni dve leti, potem se izbrišejo.

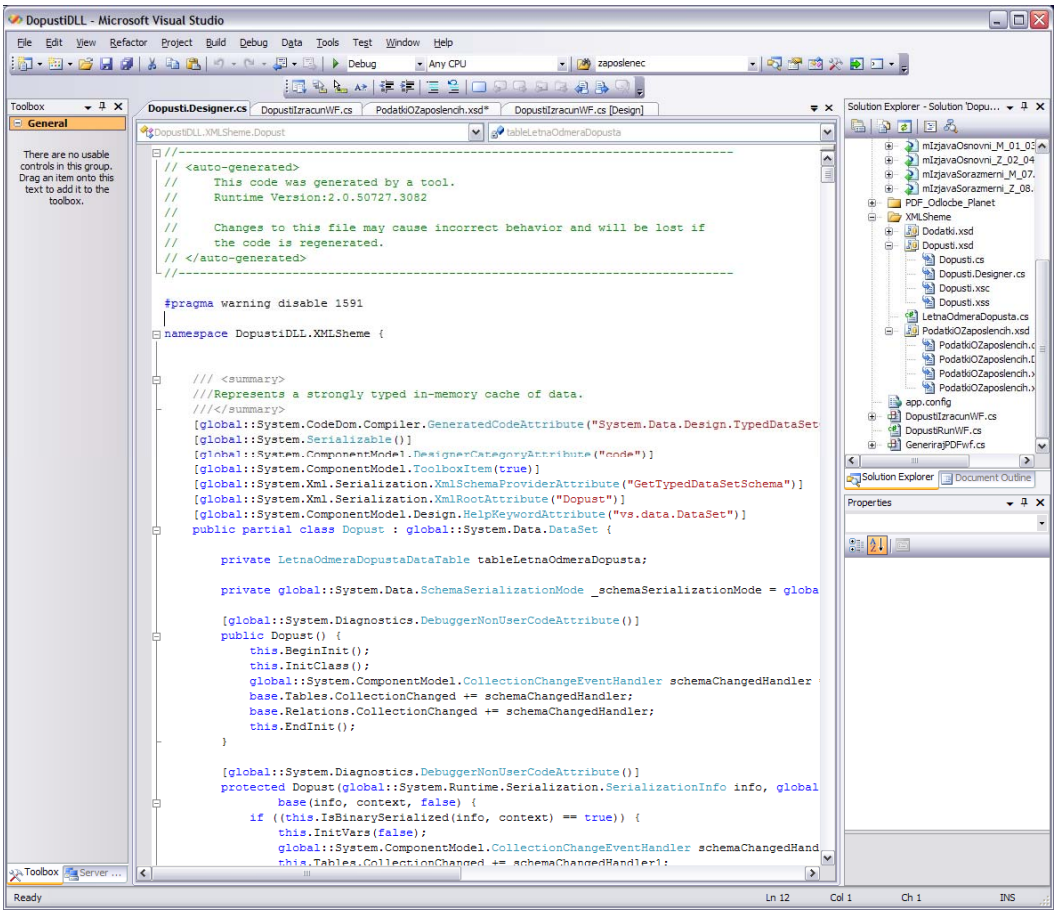
Dokumenti se shranjujejo v diskovni arhivski sistem Centera podjetja EMC. Ta sistem omogoča varno shranjevanje dokumentov po principu piši enkrat - beri večkrat (ang. WORM

– Write Once Read Many times) in ne dovoljuje spreminjanja vsebine. Takšen način arhiviranja je podprt s strani Arhiva Slovenije.

2.8 Microsoft Visual Studio 2008

Razvojno okolje Visual Studio 2008 je integrirano razvojno okolje, ki omogoča razvoj aplikacij na ogrodju.NET v jezikih, ki so z ogrodjem podprti. Ti programski jeziki so C/C++, VB.NET, C#, z dodatno namestitvijo pa se lahko uporabljajo tudi Python, F#, M, Ruby in drugi. Prav tako prepozna HTML, JavaScript in CSS. Za razvoj sistema za dopuste uporabljamo jezik C#.

Okolje omogoča razvoj okenskih, konzolnih in spletnih aplikacij, kot tudi knjižnic, spletnih storitev in drugega. Vsebuje urejevalnik izvorne kode (slika 8), ki podpira zaključevanje izvorne kode (ang. IntelliSense) in refaktoriranje. Vgrajen ima razhroščevalnik, ki deluje nad izvorno in izvedbeno kodo. Vsebuje še orodje za izgradnjo grafičnih uporabniških vmesnikov (ang. Windows Forms Designer), oblikovalca spletnih strani (ang. Web Designer), oblikovalca razredov (ang. Class Designer) in oblikovalca podatkovnih shem (ang. DataSet Designer).

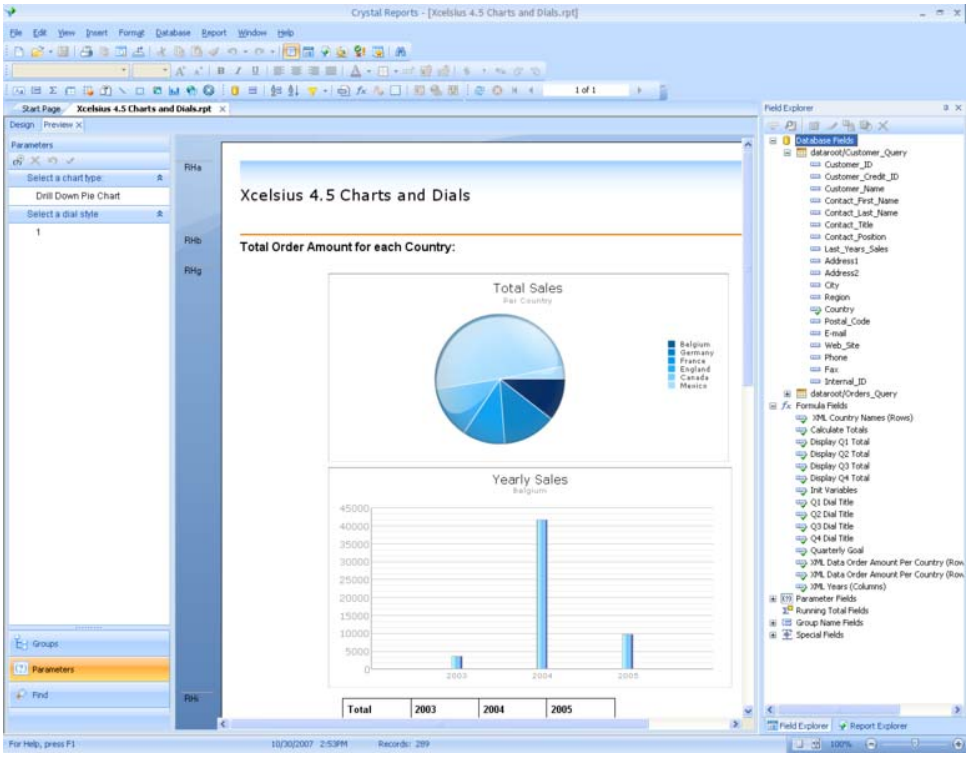


Slika 8: Razvojno okolje Microsoft Visual Studio 2008

Na voljo so tudi brezplačne različice tega razvojnega okolja z imenom Microsoft Visual Studio 2008 Express za jezike VB.NET in C#, prav tako pa je za študente na voljo brezplačna različica okolja Visual Studio 2005 Professional.

2.9 Crystal Reports

Crystal Reports je aplikacija namenjena oblikovanju in generiranju dokumentov iz širokega izbora podatkovnih virov. Integrirana je v več razvojnih okolij, med drugim tudi v Microsoft Visual Studio 2008 (slika 9) in je plačljiva. Omogoča enostavno izdelovanje dokumentov PDF s podatki, ki so shranjeni v spremenljivkah programa ali v podatkovni bazi. Tudi spreminjanje predlog dokumentov in dodajanje spojnih polj je relativno enostavno. To pomeni, da vsakoletne spremembe sistema lahko opravijo tudi produkcijske in ne nujno samo razvojne ekipe, kar močno poenostavi vzdrževanje aplikacije.



Slika 9: Primer uporabe aplikacije Crystal Reports vgrajene v okolje Visual Studio

3 Arhitektura sistema

Programski sistem za izdelavo odločb o dopustih je splet precej različnih tehnologij in orodij. Za izračunavanje rezultatov se uporablja Windows Workflow Foundation, za izdelavo dokumentov PDF se uporabljajo Crystal Reports, za pošiljanje elektronske pošte K2.NET, vse pa povezuje izvorna koda temelječa na ogrodju Microsoft .NET. V sledečih poglavjih bomo razložili vloge vseh sodelujočih komponent, najprej pa moramo opisati kakšen problem smo sploh reševali.

3.1 Zahteve naročnika

Naročnik sistema za izdelavo odločb o dopustih je kadrovska služba. Zaposleni v Enoti informatika nudimo podporo za njihovo delovanje in informatizacijo poslovanja kadrovske službe, kar je samo del celotnega informacijskega sistema v podjetju. Na željo po prenovi smo se odzvali z novimi in celovitimi rešitvami, še predvsem, ker smo že imeli na voljo najnovejšo tehnologijo in znanje za izdelavo sodobne računalniško podprte aplikacije za odmero letnega dopusta. Rešitev problema vključuje vse sektorje znotraj informatike, zato je bilo potrebno dele povezati v celoto s pomočjo ostalih sodelavcev in aplikacij, za katere ti skrbijo. Najprej pa moramo pogledati, zakaj je do prenove sistema sploh prišlo.

3.1.1 Obstoječe stanje

Pred uvedbo novega sistema je bilo izračunavanje dopusta za podjetje z približno tisoč zaposlenimi izredno zahtevna naloga. V kadrovski službi so štirje zaposleni za ta izračun potrebovali vsaj štiri tedne. Pri računanju je pogosto prihajalo do napak. Včasih so se podatki med izračunavanjem tudi spreminjali. Izračunane vrednosti je bilo potrebno zapisati v individualne odločbe vsakega zaposlenega, vsako odločbo je nato moral lastnoročno podpisati direktor podjetja oz. pooblaščena oseba. Dokument je bilo potrebo natisniti, zložiti na tretjine, vstaviti v ovojnico, nasloviti in poslati vsakemu zaposlenemu. Ta je moral ob prejemu podpisati povratnico in jo poslati nazaj v kadrovsko službo. Kadrovska služba je kopijo dokumenta skupaj s povratnico shranila v arhiv.

Poleg navadnih odločb o dopustu je bilo potrebno ročno izdajati tudi posebne odločbe o odmeri dopusta. Te so izdajali zaposlenim, ki so prišli v podjetje po 1. januarju ali ga pred

zaključkom leta zapustili. Enak problem je nastopil ob spremembi podatkov, ob rojstvu otroka, nastopu posebnih pogojev, kot je nočno delo in podobno.

Pomagali so si z Excel-ovo preglednico, v katero so vnesli vse zaposlene in potem v stolpce vnašali vrednosti. V prvi stolpec so vnesli osnovo, v drugega število dodatnih dni zaradi otrok, v naslednjega število dodatnih dni zaradi nočnega ali izmenskega dela, dežurstva, pripravljenosti na domu, v naslednjega število dni za posebne socialne razmere ter invalidnosti ali zaradi oskrbovanja prizadete osebe. Te vrednosti so sešteli po kategorijah, ki so bile osnova dopusta, delovne razmere in socialne razmere in sešteli. Po izračunu vrednosti je bilo le-te potrebno vnesti v individualno odločbo za vsakega zaposlenega, kar je pomenilo prepisovanje "na roke". Odločbe so potem natisnili in poslali.

Zaradi zgoraj opisanih razlogov se je pojavila potreba po posodobitvi sistema, ki bi izboljšal trenutno stanje.

3.1.2 Zakonodaja

Podjetja morajo pri izračunu odmere letnega dopusta upoštevati tri merila:

- določitve zakona o delovnih razmerjih,
- določitve panožne kolektivne pogodbe,
- pogodbe o zaposlitvi oz. notranje pravilnike podjetja.

Zaradi kombinacije vseh treh določb nastanejo zelo kompleksna pravila za izračun letnega dopusta zaposlenega. Zakonodaja je podvržena pogostim spremembam, še pogosteje pa se spreminjajo kolektivne panožne pogodbe in interni pravilniki.

Po zakonu o delovnih razmerjih (v nadaljevanju ZDR), 8. poglavje, 156. člen, ima delavec v Republiki Sloveniji pravico do letnega dopusta. Ta se izračuna na podlagi različnih kriterijev. Zakon tudi določa, da je potrebno najkasneje do 31. marca v tekočem letu delavca obvestiti o odmeri letnega dopusta za tekoče leto, kot tudi, da se daljše trajanje dopusta lahko določi s kolektivno pogodbo ali s pogodbo o zaposlitvi.

Delavec pridobi pravico do celotnega letnega dopusta po šestih mesecih zaposlitve. Delavec, ki še ni zaposlen šest mesecev, ima pravico do mesečne izrabe 1/12 letnega dopusta za vsak mesec, pri čemer se najmanj polovica delovnega dneva zaokroži na cel dan dopusta.

ZDR določa, da je delavec upravičen do štirih delovnih tednov dopusta. To pomeni, da dobi delavec, ki ima pet dnevni delovni teden, 20 dni osnovnega dopusta, medtem ko dobi delavec, ki dela šest dni na teden, 24 dni osnovnega letnega dopusta. To je osnova h kateri se prištevajo dodatni dnevi glede na ostala pravila [6].

Splošna kolektivna pogodba lahko določa spremembe, vendar mora temeljiti na zakonu. Splošna kolektivna pogodba za gospodarske dejavnosti v podjetju Mobitel d.d. ne velja več, zato vsa pravila za podjetje Mobitel d.d. določata ZDR in interni pravilniki. V drugih panogah kolektivne pogodbe za gospodarske dejavnosti še vedno določajo nekatera pravila. Za naš sistem to ne bi predstavljalo problema, saj bi takšne pogoje lahko hitro vključili in upoštevali.

Pravilniki v podjetjih in pogodbe o zaposlitvi lahko določajo dodatne pogoje, ki jih je pri izračunu potrebno upoštevati. Ti so od podjetja do podjetja različni. V diplomski nalogi bomo upoštevali veljavna pravila podjetja Mobitel d.d..

3.1.3 Predlog arhitekture

V podjetju imamo že dolgoletne izkušnje z izdelavo informacijskih sistemov, tako za naše centralne dejavnosti, kot za podporo poslovanja podjetju. Hkrati smo imeli na voljo dovolj virov in sredstev ter obstoječe informacijske infrastrukture, da smo sklenili problem rešiti sami.

Podatki potrebni za izračun so že obstajali v kadrovski podatkovni bazi. V prvem koraku je bilo zato potrebno zajeti le pravila računanja. V drugem koraku smo morali sestaviti seznam različnih tipov odločb, ki jih delavec lahko dobi. Zatem pa še preučiti pošiljanje in podpisovanje dokumentov ter na koncu arhiviranje.

Zaradi pogosto spreminjajočih se pravil, je bilo potrebno zagotoviti hitro in enostavno prilagajanje sistema. Nujen pogoj je bila enostavna uporaba in preglednost pravil za računanje. Zagotoviti je bilo potrebno tudi preprosto in predvsem varno podpisovanje in pošiljanje ter na koncu elektronsko arhiviranje dokumentov. Končni rezultat naj bi zagotavljal popolno ukinitev papirnega poslovanja za namen izdaje odločb o dopustih.

Predvsem smo želeli doseči prihranke pri uporabi človeških virov, pri papirju in pošiljanju ter pri stroških arhiviranja in zmanjšati število napak.

Sklenili smo, da bomo za izračunavanje uporabili sistem poslovnih pravil (ang. Business rules), kar zagotavlja preglednost in omogoča relativno enostavno spreminjanje pravil. Hkrati

nam omogoča tudi pospešen razvoj v primerjavi z kodiranjem enakovrednega izdelka. Sam dokument bomo zapisali v elektronski obliki, kar je lahko oblika PDF ali katera druga oblika, dokumente bomo podpisovali z elektronskim podpisom temelječim na asimetrični kriptografiji, jih pošiljali po elektronski poti, obveščali po elektronski pošti in jih shranili v elektronski arhiv. Tam bodo nato vedno na voljo zaposlenim, njihovim nadrejenim, kadrovski službi in inšpekciji.

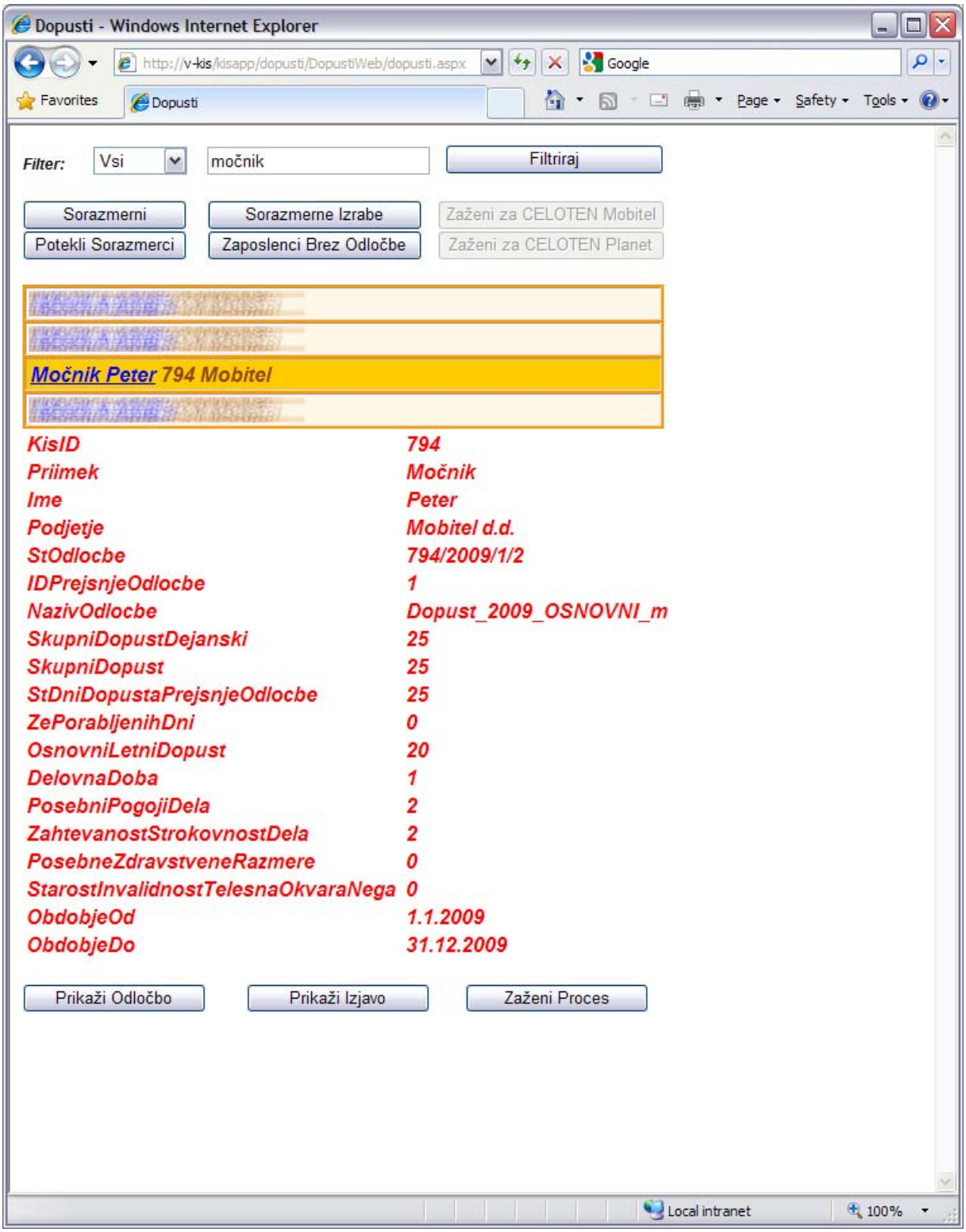
3.2 Umestitev v okolje

Zasnova sistema za elektronsko izračunavanje in izdajo odločb je sestavljena iz velikega števila različnih sodelujočih sistemov. Programska rešitev je zato precej kompleksna, ker je bilo potrebno vse dele povezati na enem mestu. To smo naredili v obliki spletne aplikacije, s katero nadzorujemo in prožimo dogodke. V tej spletni aplikaciji izberemo zaposlenega, kateremu je potrebno izračunati dopust, sprožimo sam izračun, preverimo če je izdelani dokument pravilen in ga pošljemo v postopek distribucije. Na istem mestu lahko sprožimo tudi izračun odločb za vse zaposlene v podjetju.

3.2.1 Opis aplikacije

Aplikacija za računanje dopustov je napisana kot knjižnica DLL, katere funkcije se kličejo iz spletne aplikacije. Aplikacija je sestavljena iz več delov, to so:

- spletni grafični vmesnik,
- branje podatkov,
- algoritem za računanje števila dni dopusta,
- shranjevanje rezultatov v podatkovno bazo,
- določitev predloge za dokument,
- izdelava dokumenta v obliki PDF in
- pošiljanje dokumenta v delovni tok celotnega sistema.



Slika 10: Spletni grafični vmesnik sistema

Spletni grafični vmesnik uporabljajo zaposleni v kadrovski službi za zagon aplikacije. Vsebuje filter za iskanje osebe, ki ji želijo izdati odločbo. Ko izberejo želeno osebo, kliknejo na gumb za izračun. Rezultat je viden na sliki 10. Nato lahko izberejo gumb "Prikaži odločbo", ki jim

odpre stran s predogledom odločbe o dopustu. Z gumbom "Zaženi proces" pošljejo dokument PDF v delovni tok celotnega sistema za dopuste. Izberejo lahko tudi gumb "Prikaži izjavo", kjer si ogledajo povratnico. V zgornjem delu sta še dva gumba, s katerima poženejo avtomatsko izdelavo odločb za vse zaposlene v podjetju Mobitel d.d. ali Planet9 d.o.o..

3.2.2 Podatkovni vir - KIS (Kadrovski informacijski sistem)

Zaradi posebnosti v podjetju in drugih tehtnih razlogov je kadrovski informacijski sistem podjetja Mobitel d.d. plod lastnega razvoja. Sčasoma je prerasel v kvalitetno platformo, ki je izredno prilagodljiva posebnim organizacijskim zahtevam v podjetju. Ker sistem še vedno vzdržujemo sami, lahko zelo hitro odgovorimo na zahteve uporabnikov po spremembah in prilagoditvah sistema.

Sistem temelji na relacijski podatkovni bazi MS SQL, dostop do podatkov pa je izveden na več načinov. Nekateri uporabniki uporabljajo aplikacijo v obliki debelega klienta (ang. fat client). Napisana je v jeziku Delphi, ki ima za osnovo jezik Pascal. Drugi pa dostopajo do podatkov s pomočjo spletnih aplikacij in spletnih storitev, večinoma temelječih na ogrodju Microsoft.NET.

Kadrovski informacijski sistem je zelo vpleten v podjetje. Nanj se povezuje veliko število aplikacij in sistemov, ki iz njega črpajo podatke. Nekateri sistemski dostopi imajo neposredno povezavo z bazo podatkov z uporabo Linked Serverja. Ta dostop se uporablja večinoma za branje podatkov iz strežnika. Drug način povezovanja se izvaja posredno, s pomočjo spletnih storitev (ang. web service), kot na primer interni telefonski imenik.

Mobitelov kadrovski informacijski sistem trenutno uporabljata dve podjetji, saj Mobitelova kadrovska služba opravlja storitve tudi za Mobitelovo hčerinsko podjetje Planet9 d.o.o.. Ker se odločbe za dopuste izdajajo tudi za to hčerinsko podjetje, je pomemben tudi podatek v katerem podjetju dela zaposleni in seveda organizacijska struktura tega podjetja.

Vsi podatki potrebni za izračun dopusta se nahajajo v relacijski podatkovni bazi. V isti bazi se nahajajo tudi podatki o organizacijski strukturi podjetja. Med podatki o zaposlenem in organizacijsko strukturo podjetja obstaja relacija, ki nam za vsakega zaposlenega opiše njegovo delovno mesto in enoto v kateri se to delovno mesto nahaja.

3.2.3 Opis razvitega sistema

Arhitektura rešitve bazira na ogrodju .NET in je zgrajena v tronivojski arhitekturi s strogo ločitvijo baze, aplikacijskega nivoja in grafičnega uporabniškega vmesnika. Na prvem, najnižjem, nivoju so podatkovne baze za kadrovski informacijski sistem z vsemi tabelami, poizvedbami in predvsem shranjenimi procedurami (ang. stored procedures), strežnik EMC Documentum ter strežnik K2.NET BlackPearl.

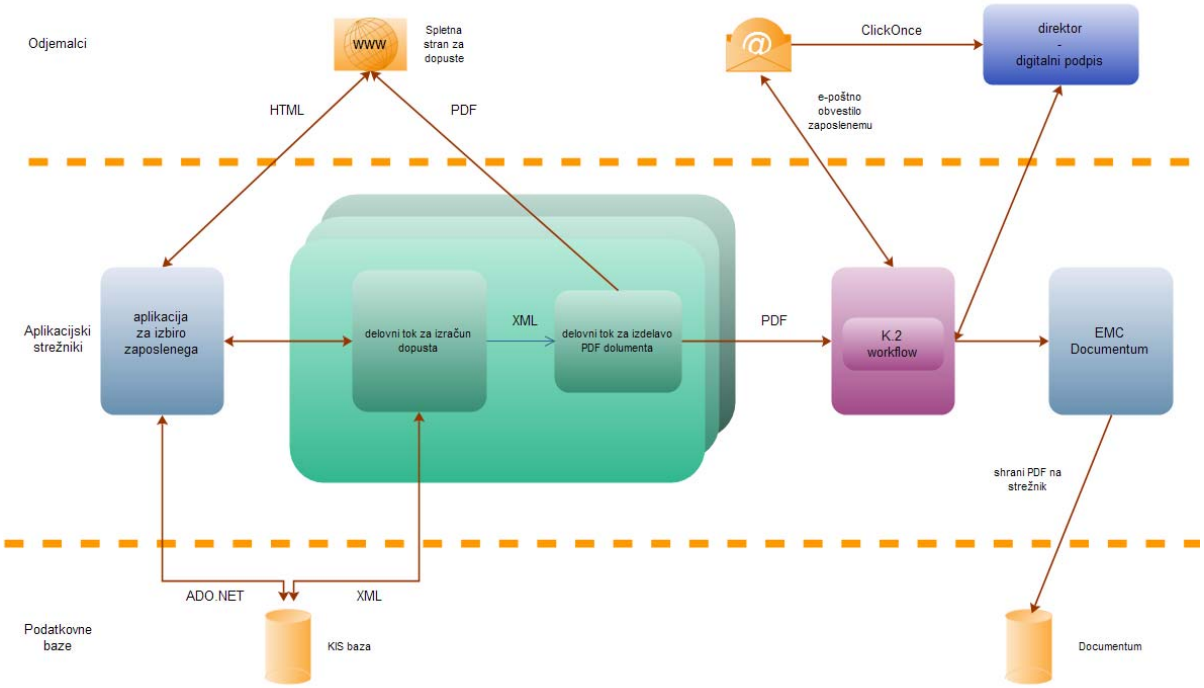
Podatkovna baza za kadrovski informacijski sistem je MS SQL Server 2005 v gruči in predstavlja močno ločitev od drugega nivoja arhitekture. Omogoča komunikacijo v obliki XML, kar je nujno potrebno za našo aplikacijo.

Na drugem nivoju delujejo Kadrovski informacijski sistem, K2.NET strežnik, digitalno podpisovanje in aplikacija za računanje dopustov.

Na tretjem, najvišjem, nivoju deluje debeli klient za Kadrovski informacijski sistem, spletni portal za dostop do aplikacije za dopuste, aplikacija za digitalno podpisovanje dokumentov MobiPodpis in poštni odjemalci.

Avtentikacija med nivoji poteka po protokolu Kerberos, ki glede na zaprto omrežje in varnostno politiko družbe zagotavlja dovolj visok varnostni nivo za potrebe podjetja.

Na sliki 11 je grafično predstavljena arhitektura celotnega sistema. Proces izdelave odločb za dopuste se prične na spletni strani, kjer uporabnik sistema z zagonom spletne aplikacije naloži seznam zaposlenih iz baze. Naloži se seznam zaposlenih s priimkom, imenom in ID-jem zaposlenih na tekoči dan. Ob izbiri zaposlenega, se pokliče shranjena procedura (ang. stored procedure) z ID-jem zaposlenega kot parametrom, ta pa nam vrne podatkovni tok v obliki XML.



Slika 11: Sistem za pošiljanje odločb o dopustih

Spletna aplikacija pokliče aplikacijo za delovne tokove s tokom XML kot parametrom, na podlagi katerega se nato izračuna dopust. Izračunane vrednosti se zaradi lažjega vpogleda v rezultate in morebitnih analiz shranijo nazaj v podatkovno bazo. Predstavljajo tudi pomemben podatek, da je bila nekomu odločba izdana. Ker je najenostavnejši način za zapisovanje v bazo kar s shranjeno proceduro iz same aplikacije, smo se odločili za tak postopek. Lahko bi jih shranili tudi s serializacijo podatkov v obliko XML in podatkovni tok nato poslali na SQL strežnik, ki bi podatke primerno pretvoril in jih shranil v tabelo.

Izhodni XML se pošlje v drug del aplikacije za dopuste, kjer se izdela dokument v obliki PDF, na podlagi podatkov iz prvega dela izračuna. Dokument PDF se kot tok bajtov (ang. byte stream) pošlje v K2.NET proces, skupaj z nekaterimi parametri, ki so potrebni za zagon delovnega toka K2.NET.

V delovnem toku sistema K2.NET se dokument najprej pošlje v podpis direktorju podjetja ali drugi pooblaščeni osebi, za kar se uporabi aplikacija MobiPodpis. Nato se pošlje zaposlenemu elektronska pošta s povezavo na dokument, ki se je pred tem že shranil v dokumentni sistem. Zaposleni s klikom na povezavo dokument odpre in ob strinjanju z vsebino podpiše povratnico. Za podpisovanje s strani zaposlenega se prav tako uporabi sistem MobiPodpis.

S tem je proces zaključen. Če želi zaposleni dokument ponovno pogledati, najde povezavo nanj v t.i. Osebnem portalu internega omrežja. Referent v kadrovski službi prejme seznam podpisanih in zavrnjenih povratnic, da lahko ukrepa naprej.

4 Implementacija

Ker se s časom podatki spreminjajo, uporabljamo poizvedbe, ki nam ob izračunu vračajo aktualne podatke za leto v katerem računamo dopust. Ob trenutku izračuna dopusta za vse zaposlene tako dobimo podatke, aktualne na dan izračuna. Za zaposlene, pri katerih pride do spremembe podatkov in s tem dopusta, se dopust ponovno izračuna ob nastopu spremembe, kar velja tudi za zaposlene, ki so v podjetje prišli po dnevu, ko je bilo za ostale zaposlene računanje dopusta že opravljeno.

4.1 Branje XML

Da bi bilo delovanje algoritma čim bolj optimalno smo se odločili, da se bodo podatki brali iz podatkovne baze samo enkrat in sicer pred začetkom računanja. To velja tako za enega zaposlenega, kot tudi kadar računamo za vse zaposlene naenkrat. Prav tako se bodo rezultati zapisali v bazo samo v zadnjem koraku, torej po računanju.

Zaradi preglednosti smo se odločili, da bo komunikacija tekla v obliki XML. Ta oblika je zelo primerna za zapis podatkov, saj ima že v osnovi podobno definirano strukturo, kot je definirana struktura podatkov v podatkovni bazi. Visual Studio 2008 vsebuje orodje, s pomočjo katerega na izredno enostaven način izdelamo podatkovni model vhodnih podatkov (ang. DataSet) iz same strukture datoteke XML.

Za branje podatkov je v bazi narejena poizvedba, ki vrača XML neposredno iz poizvedbe v tabeli. Uporablja se posebna Microsoftova funkcija "FOR XML", v zadnji vrstici poizvedbe.

Primer poizvedbe:

```
SELECT Z.*,
   (
    SELECT D .NazivDodatka, D .DodatekOd, D .IDDodatkiSifrant
    FROM [xDodatek] AS D
    WHERE D .ID = Z.KisID FOR XML PATH('Dodatek'), ROOT('Dodatki'),
TYPE
   ),
   (
    SELECT P.Naziv, P.PorodniskaOd, P.PorodniskaDo,
P.IDPorodniskaSifrant
    FROM [xPorodniskaInNega] AS P
    WHERE P.ID = Z.KisID FOR XML PATH('Porodniska'),
ROOT('Porodniske'), TYPE
   )
```

```
FROM [xZaposlenec] AS Z FOR XML PATH('Zaposlenec'),
ROOT('Zaposlenci'), TYPE
```

Bistvena lastnost te funkcije je gnezdenje. Ker že sama oblika XML zapisa podatkov omogoča gnezdenje elementov, jo uporabimo zato, da znotraj enega samega elementa XML zapišemo več podvrstic.

Iz primera zgornje poizvedbe dobimo:

```
<Zaposlenci>
 <Zaposlenec>
 <KisID>2</KisID>
 <Priimek>Močnik</Priimek>
 <Ime>Peter</Ime>
 <Dodatki>
 <Dodatek>
 <NazivDodatka>test1</NazivDodatka>
 <DodatekOd>2005-01-01T00:00:00</DodatekOd>
 </Dodatek>
 </Dodatki>
 <Porodniske>
 <Porodniska>
 <Naziv>test2</Naziv>
 <PorodniskaOd>2005-01-01T00:00:00</PorodniskaOd>
 </Porodniska>
 </Porodniske>
 </Zaposlenec>
</Zaposlenci>
```

Primer zapisa datoteke XML za primer ene odločbe:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<PodatkiZaDopuste>
<dbo._vDopusti2009>
 <ID>2650</ID>
 <Priimek>Aatest</Priimek>
 <Ime>Aatest</Ime>
 <ActiveDirectoryUsername />
 <Spol>M</Spol>
 <RojstniDatum>1975-10-14T00:00:00</RojstniDatum>
 <DatumZaposlitve>2007-12-09T00:00:00</DatumZaposlitve>
 <DD3112Let>1</DD3112Let>
 <DD3112Mes>0</DD3112Mes>
 <DD3112Dni>23</DD3112Dni>
 <NazivPojetja>Mobitel d.d.</NazivPojetja>
 <StIzobrazbe>5</StIzobrazbe>
 <NazivDelovnegaMesta>blagajnik/-čarka</NazivDelovnegaMesta>
 <SkupajEnotaDejansko>ODDELEK CRM, SEKTOR ANALIZA TRGA IN CRM, ENOTA
MARKETING, PODROČJE PRODAJA IN MARKETING</SkupajEnotaDejansko>
 <Vodenje>0</Vodenje>
 <Status>redni</Status>
```

```
<PorabljenDopustPrejsnjiDelodajalec>6</PorabljenDopustPrejsnjiDeloda
jalec>
 <MaxSteviloDni>35</MaxSteviloDni>
 <IDEnote>662</IDEnote>
-<DODATKI>
-<Dodatek>
 <Dodatek_NAZIV>Pripravljenost na domu/Intervencije - 2 dni
dopusta</Dodatek_NAZIV>
 <Dodatek_od>2003-02-01T00:00:00</Dodatek_od>
 </Dodatek>
-<Dodatek>
 <Dodatek_NAZIV>Posebni pogoji dela - 90 točk - 1 dan
dopusta</Dodatek_NAZIV>
 <Dodatek_od>2008-02-01T00:00:00</Dodatek_od>
 </Dodatek>
 </DODATKI>
- <PORODNISKA>
- <Porodniska>
 <Porodniska_NAZIV>Pripravljenost na domu/Intervencije - 2 dni
dopusta</Porodniska_NAZIV>
 <Porodniska_od>2003-02-01T00:00:00</Porodniska_od>
 </Porodniska>
- <Porodniska>
 <Porodniska_NAZIV>Posebni pogoji dela - 90 točk - 1 dan
dopusta</Porodniska_NAZIV>
 <Porodniska_od>2008-02-01T00:00:00</Porodniska_od>
 </Porodniska>
 </PORODNISKA>
</dbo._vDopusti2009>
```

4.2 Shranjevanje rezultatov

Za shranjevanje rezultatov v podatkovno bazo smo uporabili shranjeno proceduro. Podatke ji posredujemo v obliki parametrov, ta jih nato shrani v tabelo "Dopusti".

```
CREATE PROCEDURE [dbo].[spZapisiDopust]
@ID int,
@DopustVrsta varchar(255),
@DopustTip varchar(255),
@DopustOpis varchar(255),
@Username varchar(255),
@StOdlocbe varchar(50),
@SteviloDniDopusta int,
@Opombe varchar(255),
@OpisPrint varchar(255)

AS
DECLARE @SteviloOdmer int

UPDATE Dopust
SET        VeljalDo = getDate()
```

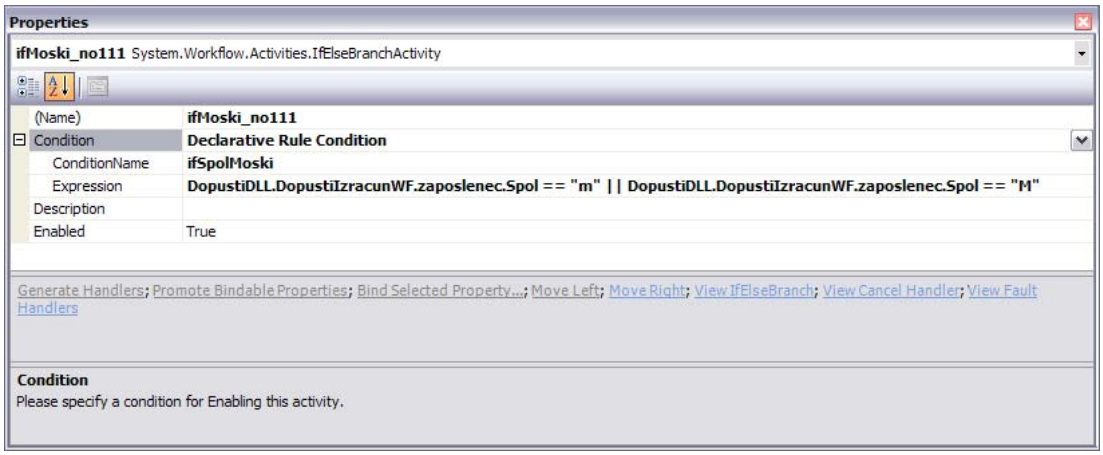
```
WHERE ID=@ID AND ZaLeto= year(getdate()) AND (DatumInCas < (getDate()
- '00:00:05.000')) AND VeljalDo IS NULL

INSERT INTO Dopust (ID, DopustVrsta, DopustTip, DopustOpis,
SteviloDniDopusta,StOdlocbe,Username, Opombe, DatumInCas, ZaLeto,
OpisPrint)
VALUES (@ID, @DopustVrsta, @DopustTip, @DopustOpis,
@SteviloDniDopusta,@StOdlocbe, @Username, @Opombe, getdate(),
year(getdate()), @OpisPrint)
```

4.3 Postopek za izračun dopustnih dni

Podatki, prebrani iz datoteke XML, se shranijo v objekt "LetnaOdmeraDopusta". Ta je definiran kot razred z atributi, kjer vsak atribut hrani vrednost za določen parameter iz baze.

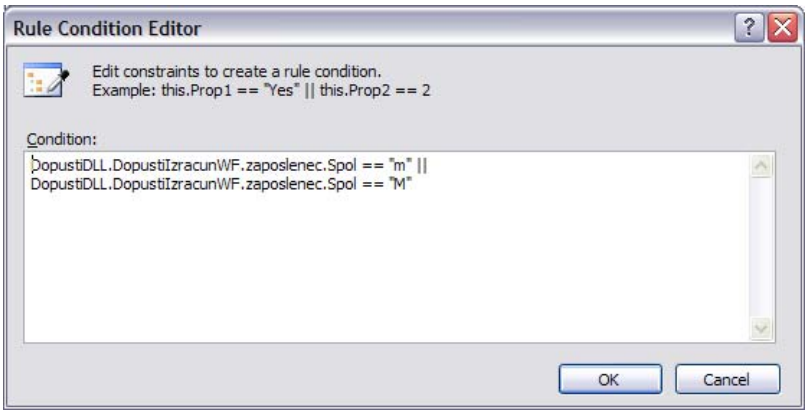
```
public class LetnaOdmeraDopusta
{ public int KisID { get; set; }
public string Priimek { get; set; }
public string Ime { get; set; }
public string ADUserName { get; set; }
public string Spol { get; set; }
public int IDPrejsnjeOdlocbe { get; set; }
      .
      .
      .
}
```

Podatki objekta "LetnaOdmeraDopusta" se vrednotijo v delovnem toku, kjer se v vsaki aktivnosti izvede primerjava pogojev. Te dodamo z izbiro pogoja za deklarativno pravilo, do česar pridemo z dvoklikom na okence aktivnosti. Lastnosti aktivnosti so prikazane na sliki 12.



Slika 12: Lastnosti aktivnosti

V polje za formulo (ang. Expression) vnesemo primerjalni pogoj v obliki izvorne kode, kar je prikazano na sliki 13.



Slika 13: Urejevalnik pogoja pravila

Ta pogoj se zapiše v datoteko z končnico .rules.

Primer zapisa s končnico .rules:

```
<RuleExpressionCondition Name="IFvodja10Ljudi">
     <RuleExpressionCondition.Expression>
    <ns0:CodeBinaryOperatorExpression Operator="ValueEquality"
xmlns:ns0="clr-namespace:System.CodeDom;Assembly=System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
  <ns0:CodeBinaryOperatorExpression.Left>
    <ns0:CodePropertyReferenceExpression PropertyName="Vodenje">
      <ns0:CodePropertyReferenceExpression.TargetObject>
        <ns0:CodeFieldReferenceExpression FieldName="zaposlenec">
          <ns0:CodeFieldReferenceExpression.TargetObject>
          <ns0:CodeTypeReferenceExpression
Type="DopustiDLL.DopustiIzracunWF, DopustiDLL, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=null" />

  </ns0:CodeFieldReferenceExpression.TargetObject>
                </ns0:CodeFieldReferenceExpression>
              </ns0:CodePropertyReferenceExpression.TargetObject>
            </ns0:CodePropertyReferenceExpression>
          </ns0:CodeBinaryOperatorExpression.Left>
          <ns0:CodeBinaryOperatorExpression.Right>
            <ns0:CodePrimitiveExpression>
              <ns0:CodePrimitiveExpression.Value>
                <ns1:Boolean xmlns:ns1="clr-
namespace:System;Assembly=mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">true</ns1:Boolean>
```

```
            </ns0:CodePrimitiveExpression.Value>
          </ns0:CodePrimitiveExpression>
        </ns0:CodeBinaryOperatorExpression.Right>
      </ns0:CodeBinaryOperatorExpression>
    </RuleExpressionCondition.Expression>
  </RuleExpressionCondition>
```

4.4 Klic metode za zagon delovnega toka

Metodo za zagon delovnega toka pokličemo tako, da pokličemo konstruktor tega objekta. Kot parameter mu podamo vhodni tok podatkov v obliki XML. Metoda po končani izvedbi prav tako vrne tok XML.

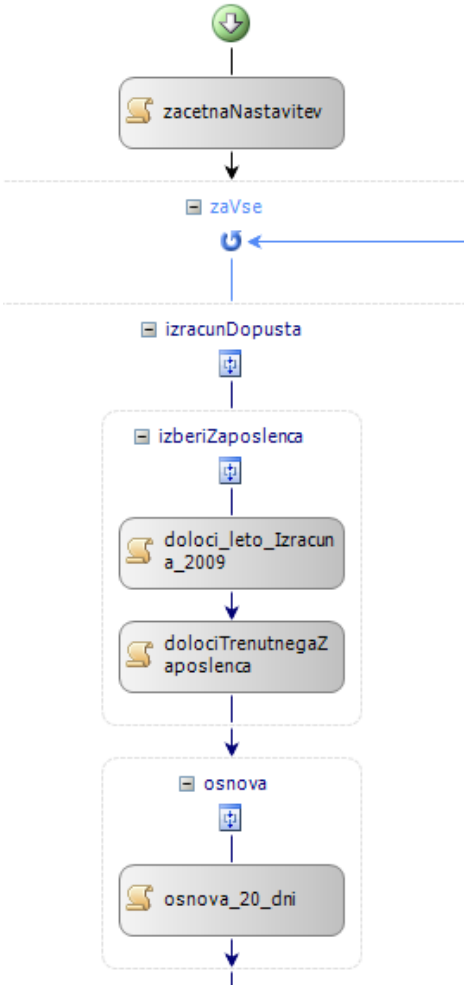
```
WorkflowRuntime workflowRuntime = new WorkflowRuntime();

AutoResetEvent waitHandle = new AutoResetEvent(false);
Dictionary<string, object> parametri = new Dictionary<string,
object>();
parametri.Add("PodatkiXML", podatkiXML as Stream);
parametri.Add("DopustiXML", dopustiXML as Stream);

workflowRuntime.WorkflowCompleted += delegate(object sender,
WorkflowCompletedEventArgs e)
 {
 dopustiXML = (Stream)e.OutputParameters["DopustiXML"];
 waitHandle.Set();
 };

workflowRuntime.WorkflowTerminated += delegate(object sender,
WorkflowTerminatedEventArgs e)
 {
 waitHandle.Set();
 };

WorkflowInstance instance =
workflowRuntime.CreateWorkflow(typeof(DopustiDLL.DopustiIzracunWF),
parametri);

instance.Start();
waitHandle.WaitOne();
```

To je praktično vsa izvorna koda, ki je potrebna za zagon delovnega toka. Vsi pogoji v delovnem toku so sicer prav tako zapisani v obliki izvorne kode, shranjeni pa so znotraj posameznih aktivnosti.

4.5 Primer izdelave odločbe

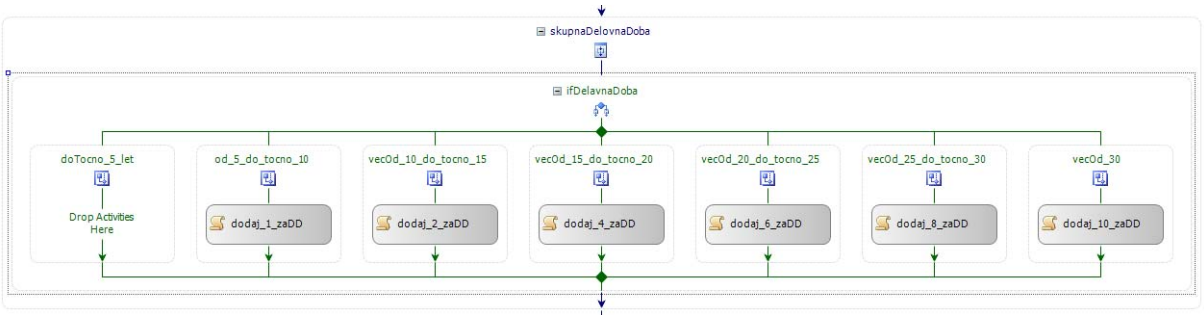
V tem poglavju bo prikazano, kako preprosto je razumeti postopek izdelave ene odločbe brez poznavanja ene same vrstice izvorne kode. Sledil bo opis vsakega posameznega koraka ter rezultat računanja. Skozi ves postopek se računajo delne vsote dopusta po kategorijah. Te se na koncu seštejejo v dejansko število dni dopusta, delne vsote pa se zaradi analiz shranijo v podatkovno bazo.

Izdelava odločbe za izračun letnega dopusta se prične z določitvijo leta izračuna in izbiro zaposlenega, kadar jih je več v vhodni datoteki. Vsi zaposleni v podjetju Mobitel d.d. imajo za osnovo 20 dni dopusta. Ta vrednost je vstavljena kot izvorna koda (ang. Code Activity), ker se lahko v naslednjem letu spremeni. Bistveno je, da je v delovnem toku vidna, ker predstavlja osnovo izračuna (slika 14).



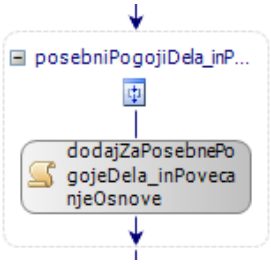
Slika 14: Začetni del delovnega toka z osnovnimi nastavitvami.

V naslednjem koraku moramo upoštevati lestvico dosežene delovne dobe. Na podlagi te lestvice se zaposlenemu dodeli dodatno število dni dopusta v odvisnosti od dopolnjene delovne dobe. Te vrednosti se skozi leta prav tako spreminjajo zaradi spreminjanja zakonodaje. Lestvica za leto 2009 je prikazana na sliki 15. Na podlagi te lestvice se k osnovi prišteje od 0 do 10 dni dopusta.



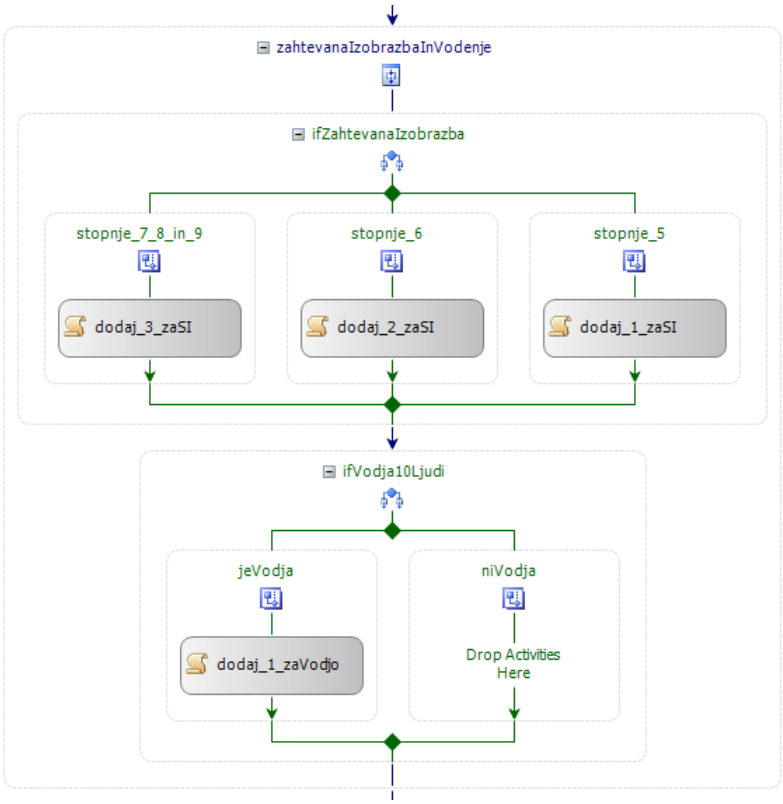
Slika 15: Lestvica dosežene delovne dobe

Pri nekaterih zaposlenih pride do povečanje števila dni dopusta zaradi posebnih pogojev dela. To so izmensko ali nočno delo, pripravljenost za delo na domu in drugi službeni razlogi. Podatek o številu dodatnih dni dobimo iz baze, tako da se vrednost prišteje k skupni vsoti. Število dodatnih dni dopusta je stvar internih pravil in za sam postopek ni pomembno (slika 16).



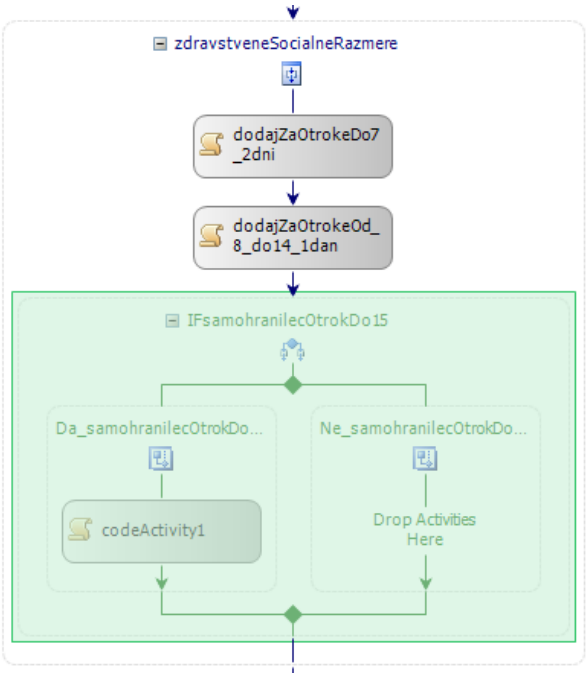
Slika 16: Povečanje osnove dopusta zaradi posebnih pogojev dela

Število dodatnih dni dopusta se poveča tudi v odvisnosti od zahtevane izobrazbe za delovno mesto. Ta ni nujno enaka doseženi formalni izobrazbi in je lahko za eno stopnjo višja od slednje. Prištejemo od enega do tri dni za V., VI. ali VII. stopnjo zahtevane izobrazbe. Dodaten dan dopusta pridobi zaposleni, ki je vodja več kot desetih zaposlenih (slika 17).



Slika 17: Povečanje osnove dopusta zaradi izobrazbe in vodenja

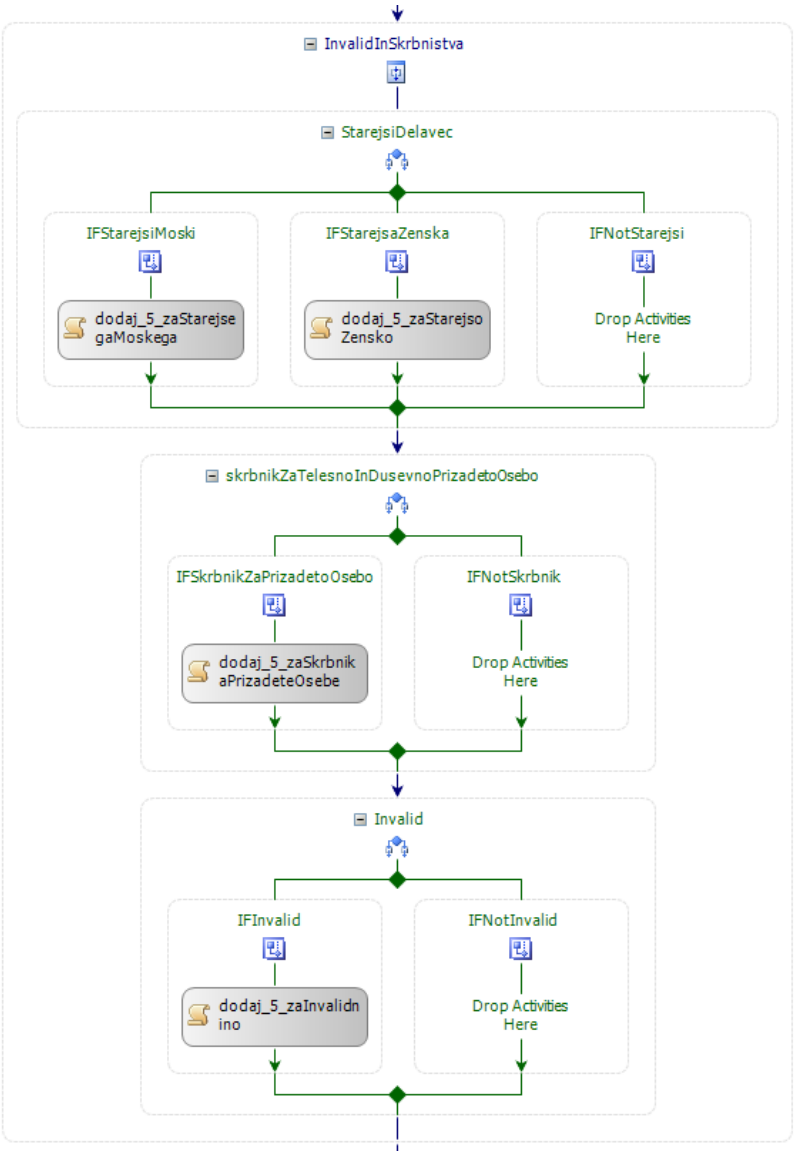
V skupini zdravstvene in socialne razmere se zaposlenemu povečuje dopust zaradi otrok. Za vsakega otroka, do sedem let starosti, pridobi zaposleni dva dni dopusta, za vsakega otroka, med sedmim in štirinajstim letom, pa en dan. Včasih je aktualno tudi vprašanje, če je zaposleni samohranilec oz. samohranilka, ker se zaradi tega lahko dopust poveča. Zato obstaja v delovnem toku tudi ta odločitev. Ker v letu izdelave sistema to ni vplivalo na izračun števila dni dopusta, je pogoj onemogočen. V delovnem toku je dodan zaradi verjetne uporabe v naslednjih letih (slika 18).



Slika 18: Sekvenca za vrednotenje zdravstvenih in socialnih razmer

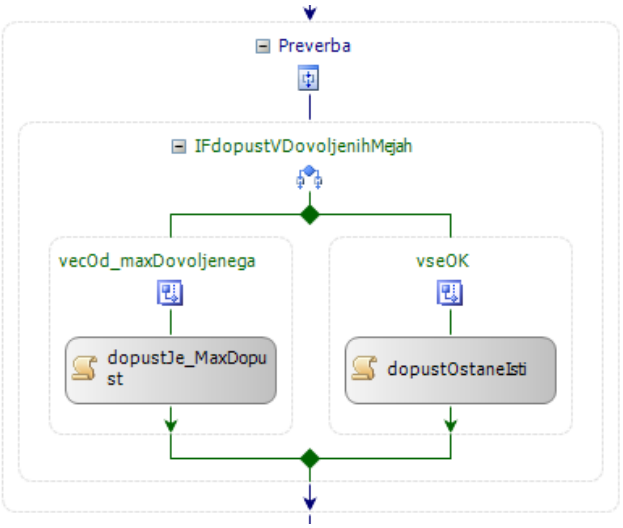
Starejšim delavcem pripada po zakonodaji še pet dodatnih dni dopusta. Pogoj za status starejšega delavca je za ženske in moške trenutno različen, v nekaj letih pa bosta pogoja izenačena. V letu 2009 moški izpolni pogoj za pridobitev statusa starejšega delavca pri 55 letih, ženske pa pri 53 letih. Vsako leto se pogoju za ženske prišteje štiri mesece. Pogoja za ženske in moške se bosta izenačila v letu 2015.

Kadar zaposleni skrbi za osebo s hujšo telesno okvaro ali duševno prizadetostjo ali ima priznano več kot 60% stopnjo invalidnosti, se mu prišteje dodatnih pet dni dopusta (slika 19).



Slika 19: Dodatni dnevi za invalidnost in skrbništvo

Ko dobimo končno vsoto dni dopusta, je le-ta lahko glede na pravila v podjetju previsoka. Ker se zgornja meja lahko spreminja, obstaja v postopku omejitev, ki določa maksimalno število dni dopusta. V letu 2009 je bila zgornja meja postavljena pri 35 dneh (slika 20).



Slika 20: Preverjanje zgornje meje doseženega dopusta.

Tako smo prišli do skupnega števila dni dopusta do katerega je upravičen vsak zaposleni.

Po izračunu števila dni dopusta se delovni tok nadaljuje z ugotavljanjem tipa predloge, ki naj se uporabi za odločbo. Večina zaposlenih bo dobila navadno odločbo o dopustu. Kar sledi iz predpostavke, da bodo vsi zaposleni, ki so bili v podjetju že pred začetkom letošnjega leta in imajo pogodbo o zaposlitvi za nedoločen čas, zaposleni vsaj do 31.12. tega leta.

Vsi ostali zaposleni, ki ne izpolnjujejo pogojev za navadno odločbo, dobijo odločbo o sorazmernem deležu dopusta in sorazmerni izrabi le-tega.

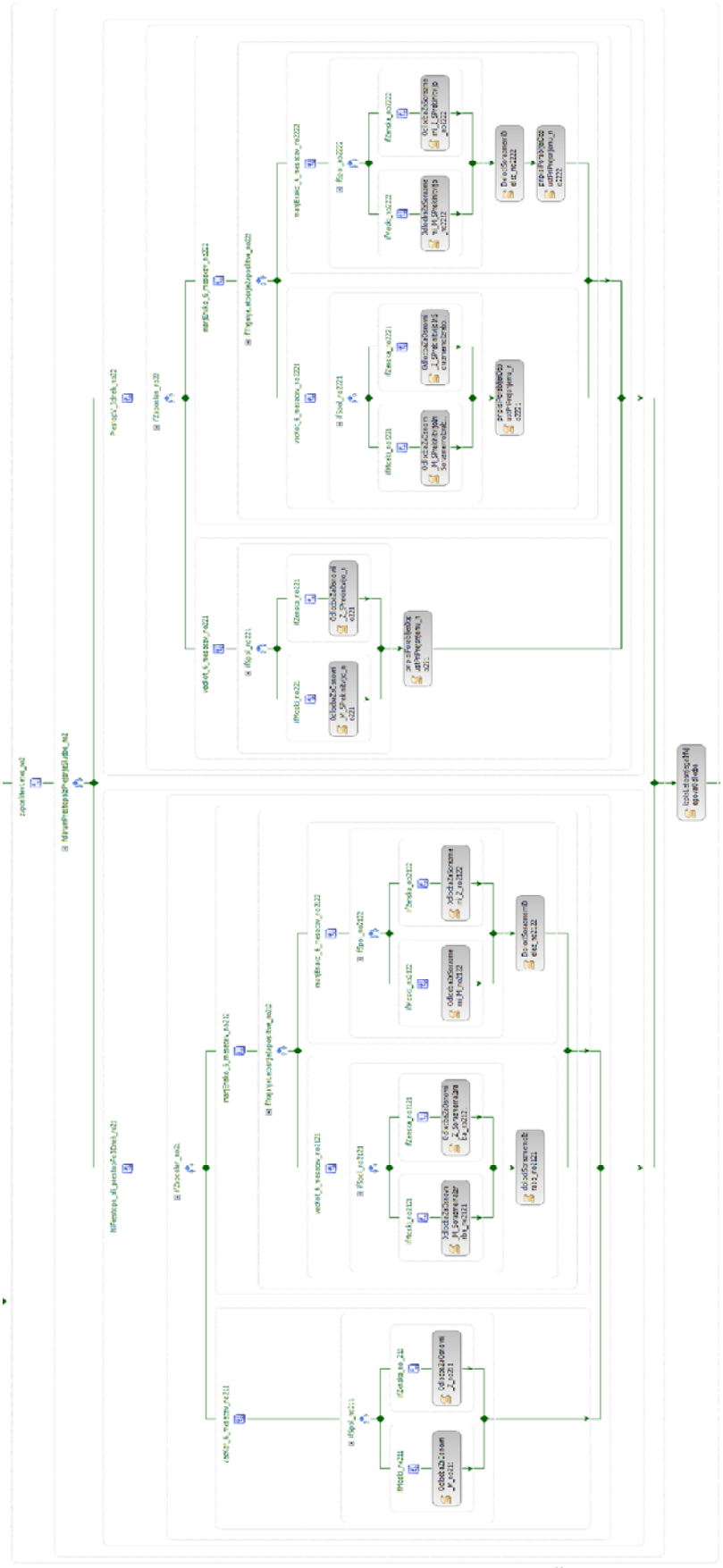
4.5.1 Sorazmerni delež izrabe letnega dopusta in druge odločbe

Zaradi velike fluktuacije zaposlenih in s tem zaposlitev, precej ljudi prejme posebno odločbo o sorazmerni izrabi dopusta.

Uporabljajo se štirje tipi odločb:

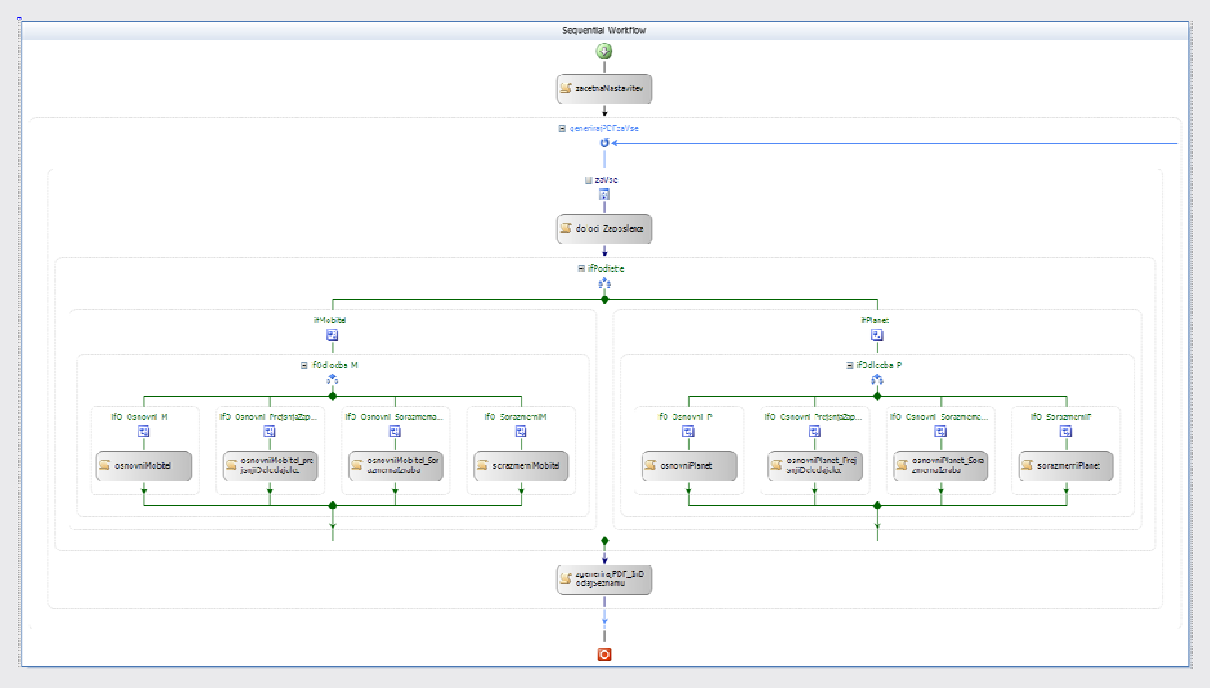
- redna odločba,
- redna odločba z prenosom dopusta od prejšnjega delodajalca,
- sorazmerna izraba dopusta, dokler zaposleni ni v podjetju vsaj pol leta,
- sorazmerni delež, ker zaposleni v tem letu ne bo delal več kot pol leta.

Tip odločbe se določi na podlagi vhodnih podatkov. Delovni tok za izbiro odločbe ugotovi katera je primerna za konkretnega zaposlenega (slika 21).



Slika 21: Ugotavljanje tipa predloge za odločbo.

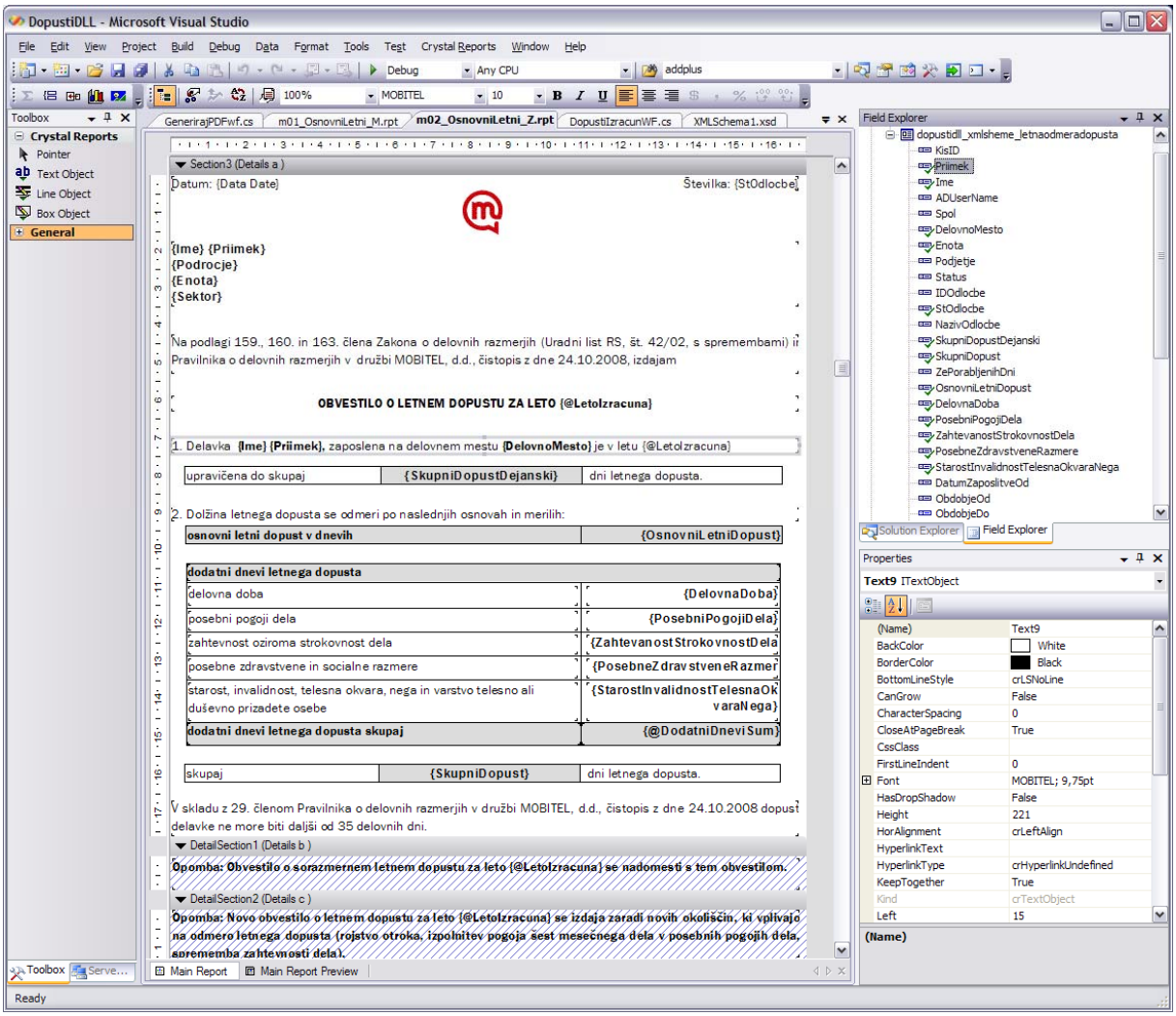
Predloge za odločbe so narejene za oba spola, zato so podvojene s spremenjenim besedilom v odvisnosti od spola zaposlene osebe. Ko je ugotovljeno katera odločba je primerna (slika 22), se pokliče metodo narejeno z orodjem Crystal Reports.



Slika 22: Diagram procesa za izbiro primerne predloge

4.5.2 Generiranje dokumentov z pomočjo sistema Crystal Reports

Dokumente je potrebno zapisati v obliko PDF, ki je splošna oblika dokumentov v podjetju. Za izdelavo dokumentov uporabimo orodje Crystal Reports za Visual Studio. S tem orodjem smo v času razvoja sistema izdelali predloge tako, da je nespremenljivo besedilo zapisano neposredno v predlogi. Na mestih, kjer se pojavljajo izračunane vrednosti dopustov, smo vstavili spremenljivke, katerih vrednosti se izpolnijo ob izdelavi dokumenta (slika 23).



Slika 23: Crystal Reports vgrajeni v okolje Visual Studio

Dokument, katerega primer vidimo na sliki 24, se nato kot tok bajtov (ang. ByteStream) pošlje v delovni tok K2.NET.

Datum: 04.06.2009 Številka: 794/2009/1

Peter Močnik
SEKTOR PODPORA DELOVANJU DRUŽBE
ENOTA INFORMATIKA
PODROČJE ZA TEHNOLOGIJO

Na podlagi 159., 160. in 163. člena Zakona o delovnih razmerjih (Uradni list RS, št. 42/02, s spremembami) in Pravilnika o delovnih razmerjih v družbi MOBITEL, d.d., čistopis z dne 24.10.2008, izdajam

OBVESTILO O LETNEM DOPUSTU ZA LETO 2009

1. Delavec **Peter Močnik,** zaposlen na delovnem mestu **razvijalec/-ka analitik/-čarka** je v letu 2009

upravičen do skupaj	**25**	dni letnega dopusta.

2. Dolžina letnega dopusta se odmeri po naslednjih osnovah in merilih:

osnovni letni dopust v dnevih	**20**

dodatni dnevi letnega dopusta	
delovna doba	**1**
posebni pogoji dela	**2**
zahtevnost oziroma strokovnost dela	**2**
posebne zdravstvene in socialne razmere	**0**
starost, invalidnost, telesna okvara, nega in varstvo telesno ali duševno prizadete osebe	**0**
dodatni dnevi letnega dopusta skupaj	**5**

Skupaj	**25**	dni letnega dopusta.

V skladu z 29. členom Pravilnika o delovnih razmerjih v družbi MOBITEL, d.d., čistopis z dne 24.10.2008 dopust delavca ne more biti daljši od 35 delovnih dni.

IZRABA LETNEGA DOPUSTA

Letni dopust lahko delavec izrabi v več delih s tem, da mora en del trajati najmanj dva tedna, ki ju mora delavec izrabiti do konca tekočega leta. Preostanek letnega dopusta lahko delavec izrabi do 30. junija naslednjega koledarskega leta.

PRAVNI POUK:

V kolikor delavec meni, da so mu bile s tem aktom kršene pravice iz delovnega razmerja, lahko pisno zahteva od delodajalca, da kršitev odpravi.

Vročiti: - delavcu,
- Kadrovskemu sektorju.

Klavdij Godnič
Glavni izvršni direktor

Digitalni podpis:
Novinšek Petrovcic Tanja
05.jun.2009 08:13:25 +02:00
Podpis
Mobitel, d.d.

Slika 24: Končna oblika dokumenta PDF z vgrajenima podpisoma

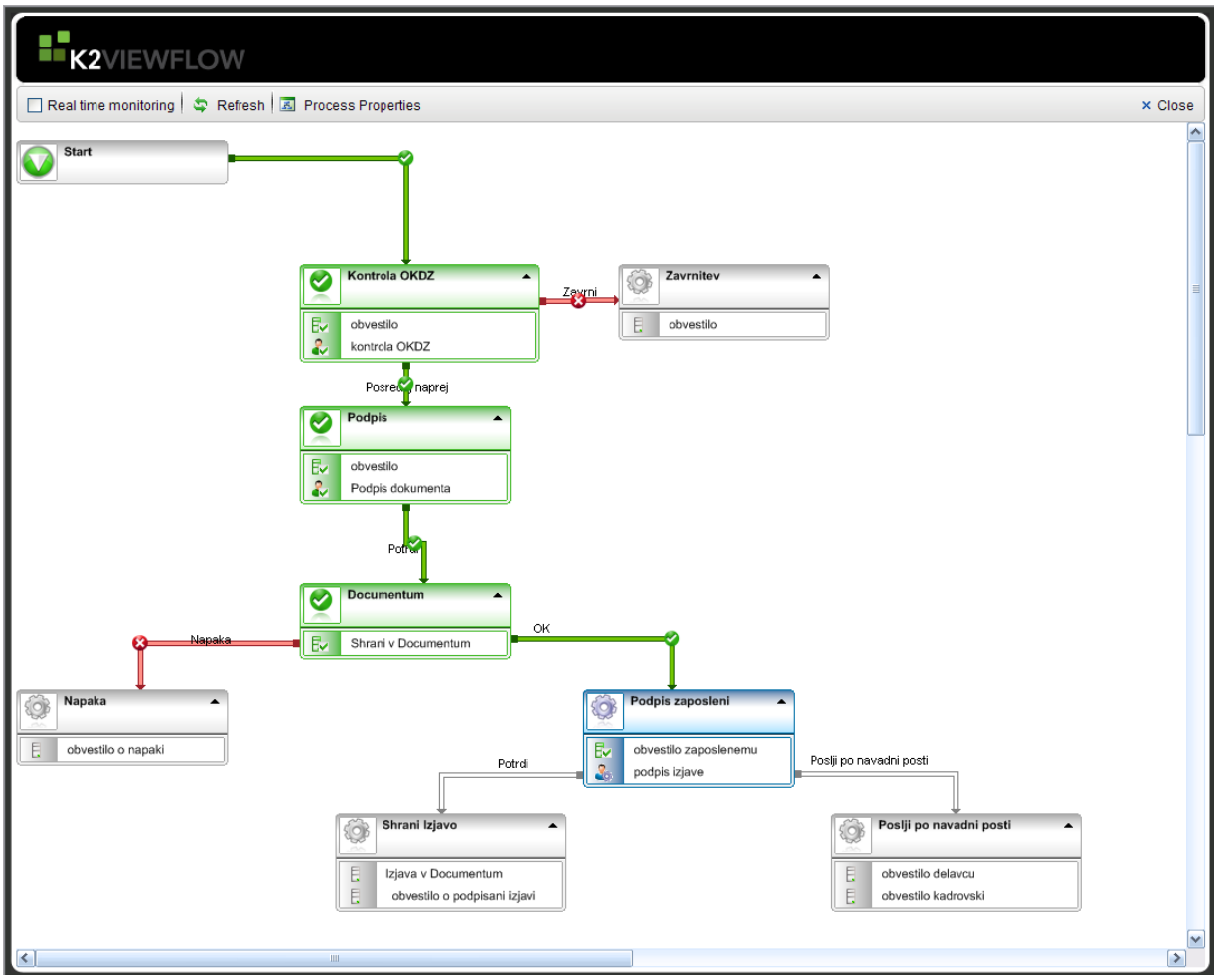
4.5.3 MobiPodpis, K2.NET proces, EMC Documentum

Naloga aplikacije za podpisovanje dopustov se konča, ko pokliče sistem za postopke. Zažene se ga z izvorno kodo:

```
Connection K2Connection = new Connection();
K2Connection.Open(_serverName, connectionString.ToString());
K2Connection.ImpersonateUser(user);
procesName = "Kadrovska\\Letna odmera dopusta";
processInstance.DataFields["ADUserName"].Value = data.ADUserName;
processInstance.DataFields["Dokument"].Value =
Session["dopustOdlocba"] as byte[]; // arrayOfByte odlocbe
processInstance.DataFields["Izjava"].Value = Session["dopustIzjava"]
as byte[]; // arrayOfByte izjave
K2Connection.StartProcessInstance(processInstance);
.
.
.
```

Uporabljeni parametri so elektronski naslov oz. uporabniško ime prejemnika dokumenta, elektronski naslov podpisnika dokumenta, naziv delovnega toka, ki naj ga uporabi in še nekateri.

V sistemu postopkov dokument sledi delovnemu toku, ki je prikazan na sliki 25.



Slika 25: Delovni tok za pot splošnega dokumenta v sistemu postopkov

Na sliki 25 je prikazan samo eden izmed več postopkov v uporabi in sicer postopek za izdajo individualnih odločb, ki jih želi referent v kadrovski službi pred pošiljanjem še preveriti. Postopek za izdajanje odločb za celo podjetje je zelo podoben, razlika je predvsem v tem, da dokumenta v kadrovski službi med postopkom ne kontrolirajo.

Dokument se najprej pošlje osebi, ki je sprožila postopek v preverjanje. Ta lahko dokument odobri ali zavrne. V primeru da je dokument odobren, se pošlje v podpis direktorju podjetja ali drugi pooblaščeni osebi. Ko ta dokument podpiše, se ga pošlje v kadrovsko mapo sistema Documentum. V naslednjem koraku dobi zaposleni elektronsko pošto z obvestilom o prejetju odločbe o letni odmeri dopusta in internetno povezavo do dokumenta. Hkrati ga v sistemu za podpisovanje počaka povratnica, ki jo prav tako elektronsko podpiše in s tem potrdi prejem odločbe o dopustu. Če dokumenta ne podpiše, ga sistem vsakih štirinajst dni obvesti po elektronski pošti, da ga čaka nepodpisan dokument, dokler ga ne podpiše.

Kadar pride do napake ali kadar dokument predolgo čaka na podpis, v kadrovski službi prejmejo obvestilo. Tako lahko zaposlenega individualno opozorijo, naj postopek s svojim podpisom konča.

5 Zaključek

Po pregledu izdelanega sistema bi se marsikdo lahko vprašal, zakaj smo se odločili za rešitev problema uporabiti delovne tokove, če lahko pridemo do iste rešitve z že uveljavljenim pisanjem izvorne kode? Navsezadnje bi lahko rekli, da smo za relativno enostaven problem izdelali preveč kompleksno aplikacijo in izkoristili preveč različnih tehnoloških rešitev.

Izkazalo se je, da je bila izdelava sistema za dopuste zelo kompleksna naloga, saj je bilo potrebno pri rešitvi upoštevati zahteve več strani. Najprej smo naleteli na zahteve kadrovske službe, ki je kot naročnik rešitve želela imeti na najbolj optimalen način rešen problem. Hkrati je delovanje celotnega sistema podvrženo strogi in pogosto spreminjajoči se zakonodaji. Ta narekuje hiter odziv kadrovske službe pri spremembah postopka za izdajanje odločb. Na drugi strani smo bili razvijalci z inženirskim pogledom na svet, ki smo kot običajno želeli problem rešiti s klasičnim načinom programiranja. Navsezadnje so tukaj tudi vsi zaposleni v podjetju, ki morajo pravočasno oz. celo čim prej ob začetku leta dobiti odločbo iz katere izvedo, do koliko dni dopusta so upravičeni.

Če bi želeli zadovoljiti vse vidike tega problema s klasičnim načinom pisanja aplikacij, bi prišli do trka med naročnikom, to je kadrovsko službo, in izvajalcem rešitve, torej programerjem in arhitektom rešitve.

Delovni tokovi so nam ponudili neke vrste most med obema svetovoma, ki sta sicer tradicionalno neskladna. Za programerje je bila dodana vrednost takšnega načina dela v tem, da se nam ni bilo treba ukvarjati s samim postopkom, ampak le s kodiranjem aplikacije. To pomeni, da smo se lahko pri delu osredotočali na druge težave in njihove rešitve v času, ki nam je bil na voljo.

Pri naročniku so prednosti očitne. Zaposleni v kadrovski službi ne potrebujejo nobenega posebnega znanje informatike, kaj šele programiranja, da bi lahko razumeli delovni tok in njegovo vsebino. Ob začetku koledarskega leta, ko se bodo spet postavljala pravila za tekoče leto, si bodo natisnili diagram in vizualno preverili delovni tok in nato po potrebi zahtevali spremembe. Če bi dobili v branje izvorno kodo, to ne bi bilo mogoče.

Ker je vsako aplikacijo potrebno tudi vzdrževati, se izkaže, da takšen pristop omogoča bistveno lažje prilagajanje in popravljanje aplikacije, saj nam ne glede na dokumentacijo vizualna predstavitev ponuja bistveno preglednejšo programsko kodo. Zaradi spreminjanja zakonodaje in notranje organizacije podjetja ter s tem njegovih pravilnikov, pričakujemo, da se bodo spremembe dogajale vsaj enkrat letno. Spreminjanje delovnega toka, ker je za to

uporabljeno grafično orodje bo bistveno lažje, hitrejše in narejeno z manj napakami, kot pa bi bilo popravljanje izvorne kode. Navsezadnje tudi ni verjetno, niti zaželeno, da bi razvijalci sisteme tudi vzdrževali. Tako napisane aplikacije se oddajo v skrbništvo vzdrževalni ekipi v podjetju, ki lahko spremembe izvede sama, pri čemer poznavanje celotnega sistema ni potrebno. Dovolj je obvladovanje sistema delovnih tokov. Pravila lahko spreminjajo sami po potrebi, pri tem pa se jim ni treba ukvarjati z izvorno kodo same aplikacije.

Ker je celoten postopek izredno pregleden, lahko razvijalci in naročniki ob grafični predstavitvi skupaj načrtujejo spremembe. Najlepše pri vsem skupaj je, da izvorne kode glavne aplikacije sploh ni treba spreminjati.

S prenovo sistema smo privarčevali kadrovski službi ogromno časa in ne nazadnje tudi denarja. Dokumenti so v vsakem trenutku na voljo tako kadrovski službi, kot zaposlenemu in inšpekciji ob morebitni kontroli. S sodobnim načinom poslovanja smo prenehali z uporabo papirja, kar je v obdobju povečane ekološke zavesti prijazno do okolja. Poleg tega smo zagotovili večjo varnost pri hrambi dokumentov.

Za izdelavo odločb o odmeri letnega dopusta od zdaj naprej potrebujejo en dan, namesto enega meseca. Celoten postopek lahko izvede ena, namesto štirih ali petih oseb. Seveda je potrebno kvalitetno delovanje kadrovskega informacijskega sistema na začetku procesa in ostale informacijske infrastrukture skozi celoten proces, vendar za to skrbi cela ekipa ljudi.

Zaradi modularnega načina izgradnje aplikacije lahko program uporablja samo tiste dele, ki so ob neki spremembi potrebni. Tako se pri hčerinskem podjetju družbe Mobitel, Planet9 d.o.o., uporablja samo izračun dopusta z izdelavo odločb, ne pa tudi umestitev v postopke distribucije in podpisovanja dokumenta, saj se za njihov delovni proces odločbe še vedno natisne na papir in pošlje po pošti.

Zaradi uporabe oblike XML za izmenjavo podatkov, je aplikacija neodvisna od podatkovne baze. Podatki lahko izvirajo iz baze Oracle, kot tudi iz sistema SAP, potrebno jih je le pretvoriti v obliko XML.

Aplikacijo za izračun dopustov bi lahko uporabili tudi v katerem drugem podjetju, npr. v kateri od hčerinskih družb podjetja Telekom Slovenije d.d.. Naredili bi kopijo aplikacije, v njej pa spremenili samo delovni tok po pravilih drugega podjetja, kar je relativno preprosto. Aplikacija bi prav tako vračala dokumente v obliki PDF, ki bi jih nato vročali z lastnimi sistemi.

Literatura in viri

[1] van der Aalst W. in van Hee K (2002) Workflow management: models, methods, and systems, MIT Press, Cambridge, Mass.

[2] Gačnik M (2005) Arhitektura sodobnih rešitev Microsoft .NET, Microsoft d.o.o., Ljubljana.

[3] Chappell C (2009) Introducing Windows Workflow Foundation - http://msdn.microsoft.com/sl-si/library/ee210343(en-us).aspx, str. 917 – 926, zadnji dostop december 2009.

[4] NET Framework Conceptual Overview - http://msdn.microsoft.com/en-us/library/zw4w595w.aspx, zadnji dostop december 2009.

[5] Troelsen A (2007) Pro C# 2008 and the.NET 3.5 Platform, Apress, New York USA.

[6] Zakon o delovnih razmerjih - http://www.dz-rs.si/index.php?id=101&vt=1&mandate=-1&unid=UPB|CE5E8F4BC5319735C12573A000433BC5&showdoc=1, zadnji dostop december 2009.

[7] Zakon o elektronskem poslovanju in elektronskem podpisu (ZEPEP) - http://www2.gov.si/zak/Zak_vel.nsf/0/c12563a400338836c12568fd00505349?OpenDocument, zadnji dostop december 2009.