UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Rok Doltar

Specifikacija zahtev v obliki uporabniških zgodb

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2010

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: 01695/2010
Datum: 01.09.2010

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

Kandidat: **ROK DOLTAR**

Naslov: **SPECIFIKACIJA ZAHTEV V OBLIKI UPORABNIŠKIH ZGODB**
APPLYING USER STORIES FOR REQUIREMENTS SPECIFICATION

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Opišite različne metode za izdelavo specifikacije zahtev in podrobneje predstavite možnosti, ki jih v ta namen nudijo uporabniške zgodbe. S pomočjo uporabniških zgodb nato izdelajte specifikacijo zahtev za projekt EZO (Evidenca zaprtih oseb) ter prikažite postopek načrtovanja izdaje in posmeznih iteracij pri agilnem razvoju tega projekta. Pri tem primerjajte metodo uporabniških zgodb z metodo, ki jo za specifikacijo zahtev trenutno uporablja RRC.

Mentor:

prof. dr. Viljan Mahnič

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

VSTAVI ORIGINAL IZDANE TEME s podpisom mentorja in dekana ter žigom fakultete pred vezavo

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Rok Doltar
z vpisno številko 63040026,
sem avtor diplomskega dela z naslovom:
Specifikacija zahtev v obliki uporabniških zgodb

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Viljana Mahniča
- so elektronska oblika diplomskega dela, naslov (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 18. 10. 2010 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju prof. dr. Viljanu Mahniču za izdatno pomoč pri izdelavi diplomskega dela ter sodelavcem v podjetju RRC za pomoč pri praktičnem delu na projektu EZO.

KAZALO

POVZETEK

V diplomskem delu so predstavljene metode pisanja specifikacij pri procesu razvoja programske opreme z namenom preučiti možne izboljšave zajema naročnikovih zahtev ter učinkovitosti razvoja. Podrobneje je opisana metoda uporabniških zgodb, ki je ena izmed obetavnih agilnih metod za zajem zahtev. Potek razvoja z uporabniškimi zgodbami je opisan na tipičnem projektu podjetja RRC. Primerjava interne metode podjetja RRC za zajem zahtev z metodo uporabniških zgodb je pokazala, da je pri procesu zajema zahtev oz. pisanja specifikacij možno prihraniti na času ter količini dela, vendar pa se z uporabniškimi zgodbami težje razrešujejo spori z naročnikom, kadar ta ne želi ali ne more sodelovati. Uporabniške zgodbe, kot ena izmed agilnih metod pisanja specifikacij, imajo vsekakor veliko prednosti, zato bi jih bilo smiselno uporabljati tudi v praksi ter seznaniti podjetja z novim načinom zajema zahtev.

ABSTRACT

My thesis describes a variety of methods for writing software specifications with intention to examine possible improvements of user requirements acquisition and software development efficacy. User stories, as one of the most promising among agile methods of user requirements acquisition, are described in greater detail. The course of development with user stories is described on a typical project, that I was working on in an enterprise named RRC. The comparison of the internal method of RRC enterprise and user stories shows that savings in time and amount of work are possible if we decide to switch from the currently used method to user stories, but we have more difficulties solving disagreements with our customer, when he cannot or will not participate in the process. User stories offer by all means a great deal of benefits, therefore their application in practice is reasonable and enterprises should be acquainted with this relatively new method of user requirements acquisition.

1 UVOD

Pri razvoju programske opreme se pogosto srečujemo s problemom, kako opisati, kakšnim zahtevam bo morala nova aplikacija ustrezati ter na kakšen način preveriti oz. oceniti ustreznost na novo razvite programske opreme. Poleg tega je potrebno poskrbeti za dosledno komunikacijo med vsemi osebami, ki sodelujejo pri uresničevanju določenega cilja oz. projekta. Slednji problem ni trivialno rešljiv (vsaj na večjih projektih), saj ima vsak izmed sodelujočih malo drugačne potrebe ter cilje. Vodja projekta želi učinkovito spremljati napredek, programerji si želijo dobre implementacije, preizkuševalci želijo čim lažje testiranje in ne nazadnje uporabniki želijo uporaben proizvod.

V večini primerov je potrebno najti ravnovesje med verbalno ter pisno komunikacijo med naročnikom (uporabniki) ter razvijalcem.

Pri večjih projektih se večinoma uporablja pristop k izdelavi nove programske opreme, ki poteka po naslednjih stopnjah [10]:

- Določanje uporabnikovih zahtev
  (User requirements definition)
- Določanje zahtev programske opreme
  (Software requirements definition)
- Postavitev arhitekturne zasnove
  (Architectural design)
- Podrobno oblikovanje in izdelava programske kode
  (Detailed design and production)
- Prenos programske opreme na delujoč sistem
  (Transfer)
- Uporaba programske opreme ter vzdrževanje
  (Operations and maintenance)

V diplomski nalogi se bom posvetil predvsem prvima dvema stopnjama, ki sta pri projektih, v katere je vključenih veliko oseb, zelo pomembni, predvsem pa natančno izvedeni stopnji pripomoreta k manjšemu številu težav v nadaljnjem razvoju projekta.

V teh dveh stopnjah moramo ugotoviti, kakšne so potrebe uporabnika ter posledično zahteve in omejitve, katere bo morala izpolnjevati programska oprema. Namen prvih dveh stopenj je izdelava bolj ali manj podrobnega dokumenta, ki ga imenujemo specifikacija in ga ustvarimo s sistematičnim beleženjem posameznih zahtev.

1.1 SPECIFIKACIJA

V splošnem je specifikacija eksplicitna množica zahtev, katerim mora ustrezati določen proizvod (ali storitev). Postopki za izdelavo specifikacij se lahko razvijajo interno v podjetjih, državnih organih ali organizacijah, lahko pa jih podajajo organizacije za standardizacijo (npr. ISO - International Organisation for Standardization).

Specifikacija programske opreme je (po standardu IEEE 830) popoln opis obnašanja sistema, ki ga želimo razviti. Vključuje primere uporabe, ki zaobjemajo vse interakcije, ki jih bodo uporabniki sistema imeli s sistemom samim.

Poznamo več načinov oz. metod pisanja specifikacij za programsko opremo. Na najvišjem nivoju se metode delijo na formalne ter neformalne.

Poleg natančnih metod za opis zahtev, ki jih mora programska oprema zadovoljevati, pa obstajajo številne agilne metode, pri katerih natančen opis zahtev nima najvišje prioritete pri njihovi izdelavi. Te metode se v večji meri zanašajo na verbalno komunikacijo, njihova prednost pa je tudi v možnosti lažjega prilagajanja spremembam.

Dejstvo, da je nezmožnost izdelave dobrih specifikacij glavni vzrok za neuspešnost razvoja programske opreme, je pogosto dokumentirano [6]. Med glavnimi vzroki za neuspeh najdemo:

- Nezmožnost inženirjev napisati pravilno specifikacijo zahtev.
- Želja menedžerjev zmanjšati aktivnosti na področju specifikacije zahtev, ker verjamejo, da je glavni trud potrebno vložiti v programiranje in testiranje.
- Pomanjkanje sodelovanja naročnikov pri potrjevanju pravilnosti specifikacije zahtev.
- Problemi z izbiro orodij in metodologije pri izdelavi specifikacij.

1.2 NAMEN DIPLOMSKE NALOGE

Med svojim delom v podjetju RRC na projektu EZO (Evidenca zaprtih oseb) smo se pogosto srečevali s problemi, naštetimi v prejšnjem odstavku. Ti so bili še toliko večji zaradi pomanjkanja koherence med posameznimi udeleženci, saj je bil naročnik v tem primeru državna ustanova z velikim številom zaposlenih, poleg tega pa nekateri izmed njih sploh niso imeli interesa iskati boljših programskih rešitev za procese, ki jih je potrebno izvajati v okviru zaporov oz. zavodov po Sloveniji.

Ker je ena izmed glavnih nalog inženirja reševanje problemov, sem se odločil, da se poleg samega razvoja aplikacije posvetim tudi problemom, ki niso v domeni popolnoma tehničnega znanja in obsegajo komunikacijo z naročnikom ter posredovanje in iskanje kompromisov pri nesoglasjih. Kot najpomembnejši del komunikacije obravnavam različne metode usklajevanja zahtev pri naročilih programskih rešitev - metode pisanja specifikacij. Podrobneje pa bom predstavil metodo uporabniških zgodb.

Za metodo uporabniških zgodb sem se odločil predvsem zaradi popolnoma drugačnega pristopa k zajemu zahtev, kot ga trenutno uporablja podjetje RRC in večina ostalih. Smiselno se mi zdi preučiti možnosti izboljšanja kvalitete ter hitrosti zajema zahtev, saj to pripomore k hitrejšemu razvoju ter večji uporabnosti in kvaliteti programske opreme.

V prvem delu diplomske naloge bom predstavil različne že uveljavljene metode pisanja specifikacij ter podrobneje opisal metodo uporabniških zgodb. V drugem delu pa bom predstavil in opisal specifikacije, ki so bile ustvarjene za potrebe projekta EZO, ter jih preoblikoval v ustrezne uporabniške zgodbe. Zaključni del bo posvečen primerjavi

metodologije pisanja specifikacij podjetja RRC na projektu EZO ter metodologije uporabniških zgodb.

2 PREGLED METOD PISANJA SPECIFIKACIJ

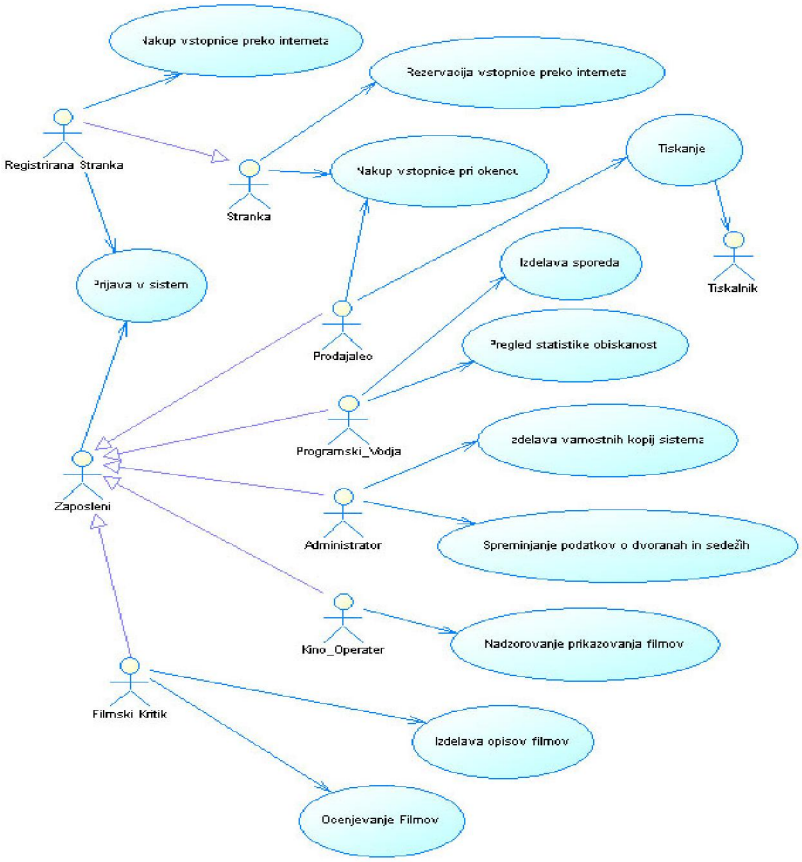
2.1 PRIMERI UPORABE (USE CASE)

2.1.1 PREGLED

Primere uporabe je kot metodo prvič predstavil Ivar Jacobson leta 1992 [11]. Primer uporabe je poenoten opis množice interakcij med sistemom in enim ali več akterji oz. uporabniki. Pri tem je akter lahko fizična oseba ali drug sistem. Poenostavljeno nam primeri uporabe opisujejo, "kdo" lahko stori "kaj" s sistemom, ki ga opazujemo. To opisujejo z uporabnikovega vidika.

Primeri uporabe so lahko napisani kot nestrukturirano besedilo, lahko pa se podrejajo omejitvam, podanim v predlogi strukture. Najbolj pogosto se uporabljajo predloge, ki jih je uvedel Alistair Cockburn leta 2001 [16].

V jeziku UML (Unified Modeling Language) nastopajo primeri uporabe v diagramu primerov uporabe, kjer so predstavljena razmerja med primeri uporabe ter akterji.



Slika 1: Primer diagrama primerov uporabe

Akterji so v podanem primeru (Slika 1) zaposleni, stranka, kinooperater itd..., primeri uporabe pa so predstavljeni v ovalnih obrobah: prijava v sistem, nakup vstopnice preko interneta, tiskanje ...

Ista fizična oseba lahko v diagramu nastopa v več različnih vlogah (akterjih) ter lahko nastopa v več različnih primerih uporabe. V našem primeru lahko zaposleni nastopa tudi v vlogi

stranke. Poleg tega lahko definiramo tudi več vrst zaposlenih (prodajalec, administrator, kinooperater, programski vodja).

Primeri uporabe namenoma obravnavajo sistem kot "črno škatlo" (black box), kar pomeni, da se osredotočajo na to, kar mora sistem opraviti in ne, na kakšen način bo naloga opravljena. Iz tega sledi, da mora primer uporabe opisovati, kaj bo sistem opravil, da bo akter dosegel cilj, pri tem pa ne sme vsebovati informacij povezanih z implementacijo ter informacij o uporabniškem vmesniku.

2.1.2 ZGRADBA

Posamezen primer uporabe opisuje, kako doseči določen cilj ali opraviti določeno nalogo. To pomeni, da je za običajne projekte potrebnih več primerov uporabe za celovit opis sistema. Stopnja formalnosti projekta ter zahtevana natančnost pogojujeta, kako podroben naj bo primer uporabe.

Čeprav ne obstaja enotna predloga za dokumentiranje primerov uporabe, so nekateri sestavni deli večinoma vključeni v posamezen primer uporabe. Med temi sestavnimi deli najdemo naslednje:

- **Ime / naslov primera uporabe (use case name)**
  Ime je unikatni identifikator primera uporabe. Sestavljala naj bi ga po en glagol oz. glagolnik in samostalnik, npr. nakup karte.
- **Verzija**
  S tem podatkom obvestimo uporabnika o stopnji razvoja tega primera uporabe, če je to potrebno.
- **Cilj**
  Opisuje namen primera uporabe. Brez cilja je primer uporabe nepotreben.
- **Kratek opis (summary)**
  Namen opisa je, da prihrani uporabniku branje celotnega primera uporabe, če želi razumeti, kaj se v primeru dogaja.
- **Akterji (actors)**
  Akter je vsaka zunanja entiteta, ki ima interakcije s sistemom. Primarni akterji so tisti ki sistem uporabljajo, sekundarni pa tisti, katere uporablja sistem.
- **Predpogoji (preconditions)**
  Vsi pogoji, navedeni v tem delu, morajo biti izpolnjeni, da se primer uporabe lahko izvede, sicer je obnašanje sistema nepredvidljivo.
- **Osnovni potek / tok dogodkov (basic course of events / primary scenario / basic flow)**
  Obvezen del vsakega primera uporabe je osnovni tok dogodkov, ki je po navadi predstavljen kot oštevilčen seznam korakov.
- **Alternativni potek / tok dogodkov (alternative paths / extensions)**
  Alternativni poteki dogodkov so variacije osnovnega poteka. Vsako testiranje pogojev v poteku ustvari nov alternativni potek. Ker število alternativnih potekov narašča hitro, je včasih potrebno predstaviti primer uporabe z diagramom aktivnosti ali diagramom poteka.

- **Popogoji (postconditions)**
  Pogoji, navedeni v tem delu, bodo veljali po zaključku primera uporabe.
- **Avtor in datum**
  V tem delu je naveden avtor in datum trenutne verzije.

Alistair Cockburn v svojem članku predlaga 3 stopnje podrobnosti pri pisanju primerov uporabe [3]:

- Kratek opis primera uporabe (use case brief), ki vsebuje dva do štiri stavke.
- Običajni primer uporabe (casual use case), ki je dolg nekaj odstavkov.
- Popoln primer uporabe (fully dressed use case), ki je formalen dokument in vsebuje zgoraj naštete sestavne dele.

2.1.3 SLABOSTI

Primeri uporabe imajo več omejitev ter slabosti. Če se odločimo za njihovo uporabo, je potrebno pregledati, ali nas bodo zanimale katere od naslednjih zahtev, ki jih primeri uporabe ne morejo zaobjeti:

- zahteve, ki ne temeljijo na interakcijah (npr. algoritem, matematične zahteve)
- zahteve, ki ne temeljijo na funkcionalnosti (npr. platforma, hitrost)

Najdemo lahko še nekatere druge slabosti:

- Primeri uporabe ne zagotavljajo jasnosti (jasnost oz. jedrnatost je odvisna od pisca in ni del predloge).
- Pri interpretiranju primerov uporabe je vedno potrebno učenje uporabnikov ter razvijalcev. Ker ni celovito določenih standardov, mora vsaka skupina postopoma razviti svojo interpretacijo.
- Uporabniki si težko vizualizirajo primere uporabe, saj ti ne vsebujejo dovolj informacij o uporabniškem vmesniku.
- Programska oprema je lahko narejena preveč dobesedno po primerih uporabe, saj zanemarjamo druge tehnike analize zahtev.

2.1.4 PRIMERI

Naslednji primer se nanaša na pisanje in pošiljanje elektronske pošte.

Naslov primera uporabe: Pisanje in pošiljanje elektronske pošte

Verzija: 1.0

Cilj: poslana elektronska pošta

Akterji: zaposleni referent

Osnovni potek dogodkov:

1. Referent izbere menijsko postavko "Novo sporočilo"
2. Sistem odpre okno "Sestavi novo sporočilo"

3. Referent napiše telo sporočila, izpolni naslov sporočila ter naslov prejemnika
4. Referent klikne na gumb "Pošlji"
5. Sistem pošlje elektronsko sporočilo

2.2 SCENARIJI

2.2.1 PREGLED

Scenarij je daljši opis predvidenih interakcij različnih vrst uporabnikov s sistemom, ki vključuje informacije o ciljih, pričakovanjih, motivih, dejanjih in odzivih. Scenarij mora odražati način uporabe sistema.

John M. Carrol predlaga naslednje karakteristične elemente scenarija [7]:

- okolje (setting)
- akterji (actors)
- cilji (goals / objectives)
- akcije in dogodki (actions and events)

Okolje predstavlja lokacijo, kjer se dogaja scenarij. Vsak scenarij mora vsebovati vsaj enega akterja. Vsak akter v scenariju poskuša uresničiti določen cilj. Carrol opredeljuje akcije in dogodke kot "zgodbo" scenarija. Akcije in dogodki so koraki, ki jih opravi akter pri doseganju cilja ali odziva sistema.

2.2.2 PRIMER SCENARIJA

Marija razmišlja o spremembi kariere. Do sedaj je bila zaposlena kot preizkuševalka programske opreme v podjetju BigTechCo. Ker je bila pred tem učiteljica matematike, se odloči, da bo srečnejša, če se vrne k poučevanju. Med brskanjem po spletu naleti na internetni portal BigMoneyJobs.com. Tu ustvari nov račun z uporabniškim imenom in geslom ter vpiše svoj življenjepis. Želi najti službo kot učiteljica matematike kjerkoli v državi Idaho, po možnosti blizu njene trenutne službe v Coeur d'Alene. Marija najde kar nekaj služb, ki ustrezajo njenim kriterijem. Najbolj jo zanima služba v North Shore School, ki je privatna šola v mestu Boise. Ker ima prijateljico Jessico, ki živi v tem mestu, vnese njen e-poštni naslov in ji posreduje povezavo do službe s pripisom, v katerem jo vpraša, ali pozna kogarkoli na tej šoli. Naslednje jutro dobi Marija e-poštno sporočilo Jessice, v katerem ji ta pove, da ne pozna nikogar na tej šoli, vendar ve, da je šola zelo ugledna. Marija se na podlagi tega odloči za novo zaposlitev in pošlje svoj življenjepis na sedež North Shore School.

2.3 IEEE 830

2.3.1 PREGLED

Računalniška sekcija (The Computer Society) združenja IEEE (Institute of Electrical and Electronic Engineers, v nadaljevanju IEEE) je objavila množico navodil za pisanje specifikacij programske opreme. [8] Ta dokument je znan pod imenom "IEEE Standard 830".

Priporočila IEEE med drugim vključujejo navodila za organiziranje specifikacijskih dokumentov in karakteristike dobrih specifikacij zahtev. Najznačilnejša lastnost specifikacij

je uporaba fraze "Sistem naj..." ("The system shall..."), ki je priporočena za pisanje zahtev funkcionalnosti.

2.3.2 PRIMER

Tipičen izsek iz specifikacije po standardu IEEE 830:

4.6. Sistem naj omogoča, da podjetje plača objavo prostega delovnega mesta s kreditno kartico.
  4.6.1. Sistem naj sprejme kartice Visa, MasterCard in American Express
  4.6.2. Sistem naj bremeni kreditne kartice pred objavo proste zaposlitve na portalu
  4.6.3. Sistem naj dodeli uporabniku unikatno potrditveno številko

2.3.3 ZGRADBA

Okvirna zgradba specifikacijskega dokumenta je sestavljena iz naslednjih poglavij:

1. Uvod (Introduction)
2. Opis izdelka (Overall description)
3. Specifične zahteve (Specific requirements)
4. Dodatni material (Additional materials)

Vsako poglavje ima več podpoglavij. Najobsežnejše in hkrati najpomembnejše je poglavje "Specifične zahteve", kjer so zahteve razdeljene v podpoglavja (npr.: zahteve funkcionalnosti, zahteve uporabniških, strojnih in programskih vmesnikov, komunikacijski protokoli, varnost, zanesljivost, prenosljivost, ...). V to poglavje spada tudi izsek iz prejšnjega primera.

2.3.4 PREDNOSTI IN SLABOSTI

Dokumentiranje zahtev do te stopnje je naporno in časovno potratno, poleg tega pa je možnost za napake velika. Velikost dokumenta v praksi doseže preko 300 strani (za srednje velik sistem). Dolžina besedila je zato lahko tudi ovira pri komunikaciji, saj ga osebe, ki bi morale biti z njim seznanjene, ne bodo prebrale v celoti, poleg tega pa jim bo iz številnih podrobnosti težje razbrati celotno sliko sistema.

Ideja napisati natančno specifikacijo za načrtovani sistem v obliki "Sistem naj..." se kljub temu zdi privlačna zaradi svoje nedvoumnosti in natančnosti. Na žalost pa je skoraj nemogoče napisati vse zahteve na ta način. Vzporedno z razvojem programske opreme se namreč vzpostavi pomembna povratna zanka, vsakič ko pride do vpogleda uporabnikov v razvijajočo se programsko opremo. Ko uporabniki vidijo program, se jim utrnejo nove ideje ali pa si celo premislijo glede starih idej. Temu se je nemogoče izogniti, z večjo natančnostjo osnovne specifikacije pa se veča tudi časovna potratnost sprememb.

Obsežnost in natančnost specifikacij pa poleg težav ob spremembah prinaša tudi nov način razmišljanja. Uveljavi se prepričanje, da je razvoj programske opreme dokončan, ko ustreza seznamu zahtev iz specifikacije, namesto da bi bil končan, ko izpolnjuje uporabnikove cilje.

Zahteve, specificirane po standardu IEEE 830, so uničile veliko projektov, ker se preveč osredotočajo na to, kar je zapisano, namesto da bi se posvečale uporabnikom. Težko je

prebrati seznam zahtev, ne da bi si v mislih avtomatsko predstavljali njihove rešitve. To si lahko ogledamo na naslednjem primeru:

4.4. Izdelek naj ima bencinski motor
4.5. Izdelek naj ima štiri kolesa
    4.5.1. Izdelek naj ima gumijasto pnevmatiko na vsakem kolesu
4.6. Izdelek naj ima volan
4.7. Izdelek naj ima jekleno ogrodje

Ko preberemo ta seznam zahtev, si skoraj zagotovo predstavljamo, da se nanaša na motorno prevozno sredstvo, verjetno avtomobil. Če pa bi nam namesto tega seznama uporabnik povedal pričakovane cilje, bi se lahko ti glasili takole:

- Izdelek mi omogoča hitro in enostavno košenje trave
- Uporaba izdelka je udobna

Ko poznamo uporabnikove cilje, dobimo popolnoma novo predstavo o izdelku. Ugotovimo, da uporabnik potrebuje kosilnico in ne avtomobila. Ta cilja iz zadnjega seznama nista uporabniški zgodbi. Namen tega primera je pokazati, da se uporabniške zgodbe za razliko od standarda IEEE 830 osredotočajo na uporabnikove želje, medtem ko je specifikacija po IEEE 830 zgolj seznam lastnosti izdelka. Zaradi tega lahko z uporabo uporabniških zgodb zasnujemo rešitev, ki bolj ustreza uporabnikovim potrebam.

3 UPORABNIŠKE ZGODBE

3.1 PREGLED

Uporabniška zgodba opisuje funkcionalnost, ki bo na voljo uporabniku ali naročniku programske opreme. Uporabniške zgodbe ne vključujejo samo pisnega dokumenta, ampak so sestavljene iz naslednjih postavk:

- pisni opis zgodbe, ki se uporablja za načrtovanje in kot opomnik
- pogovori o zgodbi, ki služijo za določanje podrobnosti
- testi, ki določajo, kdaj je zgodba zaključena

Pisni opis zgodbe se običajno zapiše s pisalom na kos papirja oz. kartico. V literaturi zasledimo izraz "Card, Conversation, and Confirmation" [9], ki na enostaven način opisuje te tri sestavne dele uporabniške zgodbe. Pri tem je potrebno poudariti, da pisni opis zgodbe ni najpomembnejši del, ampak samo simbolično uteleša naročnikove potrebe, namesto da bi jih podrobno dokumentiral.

Nekaj primerov uporabniških zgodb v projektu spletnega posredovanja zaposlitve, kot v primeru scenarija bi izgledalo takole:

- Uporabnik lahko objavi svoj življenjepis na spletnem portalu.
- Uporabnik lahko pregleduje prosta delovna mesta.
- Podjetje lahko objavi odprtje novih delovnih mest.
- Uporabnik lahko določi, kdo lahko vidi njegov življenjepis.

Vsaka uporabniška zgodba predstavlja funkcionalnost, ki bo koristna predvsem za uporabnika. Naslednje uporabniške zgodbe so primeri slabo zastavljenih uporabniških zgodb:

- Programska oprema bo napisana v jeziku C++.
- Program se bo povezoval s podatkovno bazo preko bazena povezav (connection pool).

Prva zgodba ni primerna za ta projekt, saj tu za uporabnika ni pomembno, kateri programski jezik bomo uporabili. Če pa bi bil naš produkt npr. nek API (Application Programming Interface), pa bi uporabnik lahko zapisal tako zgodbo. Druga zgodba ni primerna, ker uporabnikov ne zanimajo tehnične podrobnosti o povezovanju aplikacije s podatkovno bazo.

3.2 PODROBNOSTI V UPORABNIŠKIH ZGODBAH

Ob prebiranju primerov uporabniških zgodb se nam zastavlja vprašanje, od kod pridobiti podrobnosti v zvezi s posamezno funkcionalnostjo, ki jo zgodba opisuje. Kot primer vzemimo uporabniško zgodbo "Uporabnik lahko pregleduje prosta delovna mesta.". Če želimo začeti pisati programsko kodo in testirati rezultate, nam ta uporabniška zgodba ne zadostuje, saj potrebujemo odgovore na vprašanja, kot so:

- Po katerih kriterijih lahko uporabnik pregleduje oz. filtrira prosta delovna mesta (regija, mesto, vrsta delovnega mesta, ključne besede, ...)?
- Ali mora biti uporabnik včlanjen?
- Katere informacije se prikažejo za vsako prosto delovno mesto?

Večino teh podrobnosti lahko podamo z dodatnimi uporabniškimi zgodbami. Veliko bolje je imeti več uporabniških zgodb, ki obsegajo manjše področje, kot pa maloštevilne zgodbe, ki obsegajo večji del funkcionalnosti. Nenapisano pravilo, ki se ga izplača upoštevati, pravi, da je idealna velikost uporabniške zgodbe taka, da jo lahko en ali dva programerja sprogramirata in testirata v obdobju, ki se giblje med polovico dneva ter dvema tednoma. [2]

Kadar je uporabniška zgodba prevelika, jo je potrebno razdeliti v dve ali več manjših zgodb. Vzemimo zgodbo "Uporabnik lahko pregleduje prosta delovna mesta.". To zgodbo lahko razdelimo v tri manjše zgodbe, npr.:

- Uporabnik lahko pregleduje prosta delovna mesta po lastnostih: lokacija, višina plače, vrsta delovnega mesta, ime podjetja oz. združbe, datum razpisa.
- Uporabnik lahko pregleduje podatke o vsaki zaposlitvi, ki jo najde.
- Uporabnik lahko pregleduje podrobne podatke o podjetju, ki je razpisalo prosto delovno mesto.

Pri deljenju uporabniških zgodb na manjše enote nas ne sme zanesti v drugo skrajnost - deljenje obsega zgodb do najmanjših podrobnosti. Kot primer pretiranega deljenja na zgodbe z manjšim obsegom vzemimo zgodbo "Uporabnik lahko pregleduje podatke o vsaki zaposlitvi, ki jo najde.". Ta zgodba ima sprejemljiv obseg, zato nam je ni potrebno nadalje deliti na naslednje zgodbe:

- Uporabnik si lahko ogleda opis delovnega mesta.
- Uporabnik si lahko ogleda višino plače za delovno mesto.
- Uporabnik si lahko ogleda lokacijo delovnega mesta.

V praksi se podrobnosti na najnižjem nivoju razrešuje z razpravo med razvojno skupino in naročnikom, kar je veliko boljši pristop, kot pisanje pretirano podrobnih uporabniških zgodb. Do razpravljanja o podrobnostih pride šele takrat, ko te postanejo pomembne in ne že na začetku postavljanja zahtev.

3.3 Sprejemni testi (Acceptance tests)

Pri izdelavi uporabniških zgodb je zelo pomembno, da čim bolj točno zajamemo pričakovanja uporabnikov. Ta pričakovanja zajamemo v obliki sprejemnih testov (acceptance tests).

Sprejemno testiranje (acceptance testing) je proces potrjevanja, da je funkcionalnost zgodb razvita po pričakovanjih naročniške skupine.

Če uporabljamo papirnate kartice za zapis uporabniških zgodb, lahko na hrbtni strani zapišemo načine, na katere lahko testiramo uporabniško zgodbo. V elektronskem sistemu za vzdrževanje uporabniških zgodb te načine testiranja vnesemo za vsako posamezno zgodbo.

Zapisi o postopku testiranja naj bi bili kratki in nepopolni, poleg tega pa lahko posamezne načine testiranja dodajamo ali odstranjujemo. Smisel zapisov je v dodatnih informacijah o zgodbi, ki povedo razvijalcem programske opreme, kdaj so zaključili z razvojem sklopa, ki ga zgodba zajema. Primer seznama sprejemnih testov lahko izgleda takole:

- Preizkusi delovanje s praznim poljem za opis delovnega mesta.
- Preizkusi delovanje z zelo dolgim opisom delovnega mesta.
- Preizkusi delovanje z manjkajočim vnosom višine plače.
- Preizkusi delovanje z vnosom 6-mestne višine plače.

Obstaja več različnih tipov testiranja. Naročnik in razvojna skupina morajo sodelovati, da se zagotovi, da so načini testiranja primerni za zgodbo. Za večino sistemov je testiranje po navadi funkcionalno - zagotavlja, da aplikacija deluje (funkcionira) po pričakovanjih. Obstajajo pa še drugi načini testiranja, med katerimi najdemo [2]:

- testiranje uporabniškega vmesnika
  (Zagotavlja pravilno obnašanje komponent vmesnika.)
- testiranje uporabnosti / enostavnosti uporabe
  (Zagotavlja, da je aplikacija prijazna in uporabna za uporabnika.)
- testiranje zmogljivosti oz. kapacitete
  (Ugotavlja, kako dobro se aplikacija obnaša pod obremenitvijo.)
- testiranje ekstremnih vrednosti
  (Ugotavlja, kaj se zgodi ob vnosu pretiranih vrednosti s strani uporabnika.)

Sprejemni testi dajejo programerjem veliko informacij, ki jih lahko uporabijo že pred samim razvojem zgodbe. Recimo, da imamo zapisan naslednji sprejemni test:

- Testiraj z različnimi vrednostmi nakupa (vključno s tistimi, ki so višji od limita kartice).

Če ta test zapišemo, preden programer začne z razvojem zgodbe, se bo spomnil, da mora obravnavati primere, ko je nakup zavrnjen zaradi nezadostne plačilne zmožnosti kupca. Če tega testa razvijalec pred začetkom ne vidi, je možnost, da bo omenjen scenarij spregledal veliko večja.

3.4 FitNesse

FitNesse je računalniško podprta rešitev, napisana v programskem jeziku Java, ki omogoča avtomatizirano izvajanje sprejemnih testov. Ogrodje (framework) vsebuje strežnik z vgrajenim sistemom wiki, ki omogoča enostavno dodajanje ter urejanje večjega števila medsebojno povezanih spletnih strani z uporabo poenostavljenega označevalnega jezika (markup language).

FitNesse omogoča uporabnikom razvite aplikacije vnos vhodnih podatkov. Ti se nato interpretirajo ter uporabijo pri izvedbi sprejemnih testov. Po izvedbi testov nam FitNesse vrne povratne podatke. Prednost takega pristopa je zelo hitra povratna informacija za uporabnike, poleg tega pa si z njim pomagajo tudi razvijalci.

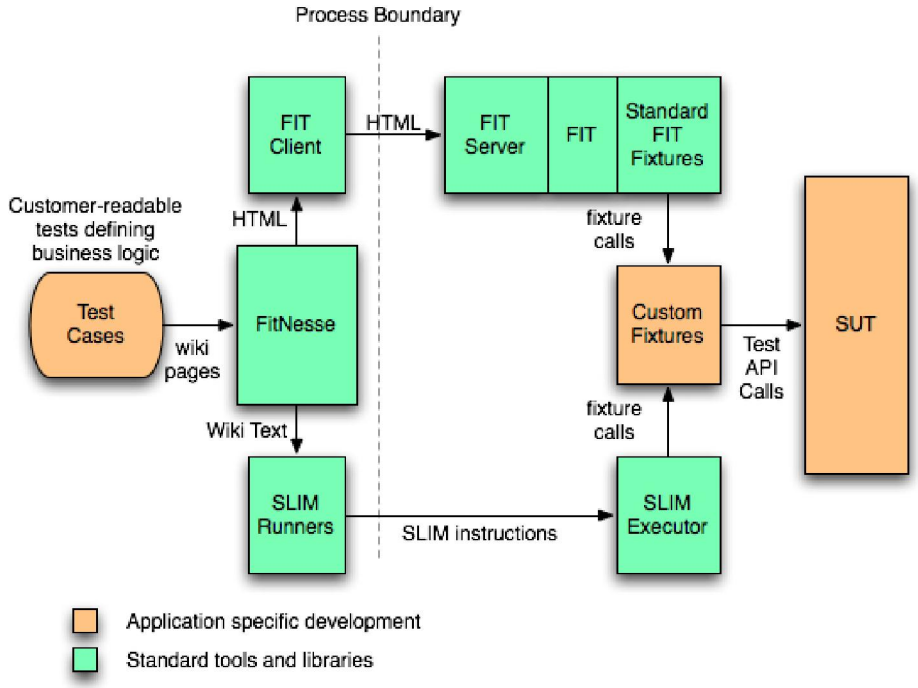
Sprejemni test v okolju FitNesse izdelamo po sledečem postopku. Najprej dodamo novo stran ali izberemo obstoječo, na kateri bo testna tabela. Testna tabela vsebuje vhodne podatke ter pričakovane rezultate oz. povratne podatke. Tabelo ustvarimo s pomočjo wiki označevalnega jezika, lahko pa jo uvozimo iz drugih programov (npr. Excell). Vsaka tabela ima svoje ime, ki

je enako imenu razreda, ki ga morajo implementirati razvijalci (fixture) in mora vsebovati potrebne metode, da se vhodni podatki ter rezultati pravilno interpretirajo. Vsaka vrstica v tabeli pa predstavlja nabor podatkov za izvedbo posameznega testa.

Ko na ustvarjeni testni strani, ki vsebuje testno tabelo kliknemo na gumb "Test", FitNesse najprej pošlje testno tabelo testnemu sistemu Slim (FitNesse je namreč uporabniški vmesnik, zgrajen na sistemu Slim). Slim poišče in zažene razred, ki ustreza testni tabeli. Ta razred kliče ustrezen del aplikacije, s čimer se izvede del programske kode, ki ga želimo testirati, nato pa vrne rezultate. Ko FitNesse prejme povratne podatke, nam za vsak test v tabeli pove, ali je bil uspešno izveden. Neuspešno izvedeni testi se obarvajo rdeče.

FitNesse je zasnovan kot ovojnica (wrapper) za orodje FIT (Framework for Integrated Test), ki izvaja testiranje sistema (SUT - Sistem Under Test). Komunikacija med obema plastema poteka preko komponente FitServer. Zaradi take zasnove celoten sistem ni omejen na aplikacije, razvite v Javi, ampak podpira različne programske jezike. Za vsak jezik potrebujemo ustrezno implementacijo orodja FIT ter FitServer. Arhitektura celotnega ogrodja je prikazana na spodnji sliki (Slika 2). Trenutno so podprti naslednji programski jeziki:

- Java
- .NET
- C++
- Delphi
- Python
- Ruby
- Smalltalk
- Perl



SLIKA 2: ARHITEKTURNA ZASNOVA SISTEMA FITNESSE

3.5 PROCES RAZVOJA Z UPORABO ZGODB

3.5.1 NAROČNIŠKA SKUPINA (CUSTOMER TEAM)

Idealno bi bilo, če bi imeli za vsak projekt samo eno osebo, ki bi razvrščala delo razvijalcev po pomembnosti, odgovarjala na njihova vprašanja, pisala uporabniške zgodbe, poleg tega pa bi bila tudi končni uporabnik programske opreme. Ker to (vsaj za večje projekte) ni mogoče, je potrebno oblikovati skupino (customer team), ki bo poskrbela, da bo programska oprema zadovoljila potrebe bodočih uporabnikov. Ta lahko vključuje preizkuševalce, upravitelje projekta in končne uporabnike.

3.5.2 POTEK RAZVOJA

Projekt, pri katerem uporabljamo uporabniške zgodbe ima drugačen potek razvoja, kot ga poznamo pri tradicionalnem kaskadnem (waterfall) modelu, pri katerem si zaporedno sledijo faze pisanja in analiziranja zahtev, snovanja rešitve, izdelovanja programske kode in testiranja. Pri kaskadnem modelu so uporabniki vključeni v proces razvoja večinoma samo na začetku (pri določanju zahtev) in ob zaključku (potrditev, da aplikacija deluje v skladu z zahtevami). Slabost modela pa je, da so uporabniki minimalno vključeni v vmesnih fazah razvoja.

Pri razvoju z uporabo zgodb morajo biti uporabniki oz. naročniki programske opreme vključeni v razvoj skozi celoten proces. To velja tako za razvoj z uporabo ekstremnega programiranja (extreme programming) [12] ali kakega izmed agilnih vrst razvoja, kot je Scrum [13]. V vseh primerih se od stranke pričakuje aktivno sodelovanje v vseh fazah razvoja.

Pri procesu izdelovanja zgodb je potrebno najprej določiti različne tipe uporabnikov sistema. Če izdelujemo spletno stran za rezervacijo potovanj z letalom, imamo lahko npr. tipe uporabnikov: pogosti uporabniki, sezonski uporabniki, enkratni uporabniki, itd. Skupina, ki skrbi za zadovoljevanje potreb uporabnikov (customer team), mora vsebovati čim več tipov uporabnikov.

Pri tem se lahko vprašamo, zakaj mora pisati zgodbe naročnik oz. naročniška skupina (customer team) in ne razvijalci programske opreme. Prvi razlog je, da mora biti vsaka zgodba napisana v jeziku poklica oz. stroke naročnika in ne v tehničnem žargonu, ki ga uporabljajo programerji. Tako lahko naročnik oz. naročniška skupina razvrsti zgodbe po prioriteti za njihovo vključevanje v iteracije in izdaje (release). Drugi razlog se skriva v tem, da je naročnik pobudnik ter primarni snovalec vizije projekta in je zato najbolj primeren za opisovanje obnašanja produkta.

Začetna množica zgodb za projekt po navadi nastane na delavnici oz. brainstormingu, kjer vsi udeleženci poizkušajo napisati čim več uporabniških zgodb. Zgodbe se lahko po potrebi dodaja tudi med razvojem. Naročniška skupina in razvijalci nato izberejo dolžino posamezne iteracije razvoja (1 - 4 tedne), ki bo ostala enaka skozi celoten razvoj ter količino dela, ki ga lahko opravijo v posamezni iteraciji. To količino dela je na začetku težko oceniti, vendar se sčasoma ocene približajo realnosti. S tema podatkoma lahko ocenimo koliko iteracij bomo potrebovali.

Za načrtovanje izdaje razvrstimo uporabniške zgodbe v množice, izmed katerih vsaka predstavlja po eno iteracijo. Velikost množic ne sme presegati ocenjene količine dela, ki ga lahko opravimo v posamezni iteraciji. Zgodbe z najvišjo prioriteto se nahajajo v prvi množici in so razvrščene po padajoči prioriteti vse do zadnje množice oz. iteracije. Zadnja iteracija je bodisi določena s časovnim rokom projekta ali z dovolj dobro zadovoljitvijo zahtev za izdajo. Ker imamo z vsako iteracijo na voljo boljšo oceno hitrosti dela, lahko prilagodimo velikosti množic za naslednje iteracije v skladu z novimi ocenami.

3.5.3 PLANIRANJE IZDAJ IN ITERACIJ

Planiranje izdaj se nanaša na določanje razmerja med časovnim tokom projekta in željeno funkcionalnostjo. Planiranje iteracij pa pomeni izbiranje uporabniških zgodb, ki bodo vključene v trenutno iteracijo.

Kot že omenjeno, je potrebno najprej razvrstiti zgodbe po prioriteti. Pri tem se upoštevajo naslednji kriteriji:

- zaželjenost funkcionalnosti pri veliki podskupini uporabnikov
- zaželjenost funkcionalnosti pri majhnem številu zelo pomembnih uporabnikov
- kohezivnost zgodbe v odnosu do drugih zgodb (Npr. zgodba "Uporabnik lahko poveča velikost slike" v aplikaciji za obdelavo slik sama po sebi nima visoke prioritete, vendar jo pridobi zaradi visoke prioritete zgodbe "Uporabnik lahko zmanjša velikost slike".)

Pri določanju prioritete zgodb je potrebno upoštevati tudi njihovo ceno. Cena zgodbe je približna in jo določijo razvijalci. Podana je s številom točk (story points), ki predstavlja velikost in kompleksnost zgodbe v odnosu do drugih zgodb. Npr. zgodba, ki ima 4 točke, naj bi bila dvakrat kompleksnejša od zgodbe z 2 točkama. Razvijalci lahko sedaj s pomočjo točk izrazijo količino dela, ki jo lahko predvidoma opravijo v eni iteraciji. Naročniška skupina nato dodeli zgodbe iteracijam tako, da število točk v eni iteraciji ne presega zmožnosti razvijalcev.

Kot primer imamo v prvi tabeli seznam vseh zgodb, razporejenih po padajoči prioriteti, v drugi tabeli pa so zgodbe razporejene v posamezne iteracije, pri čemer smo upoštevali zgornjo mejo opravljenega dela v iteraciji 13 točk.

Naslov zgodbe	Št. točk
Zgodba A	3
Zgodba B	5
Zgodba C	5
Zgodba D	3
Zgodba E	1
Zgodba F	8
Zgodba G	5
Zgodba H	5
Zgodba I	5
Zgodba J	2

TABELA 1: SEZNAM ZGODB IN NJIHOVIH CEN

Iteracija	Zgodbe	Točke
1. Iteracija	A, B, C	13
2. Iteracija	D, E, F	12
3. Iteracija	G, H, J	12
4. Iteracija	I	5

TABELA 2: NAČRT ITERACIJ ZA IZDAJO (RELEASE)

3.6 IZDELOVANJE ZGODB

Dobro napisane uporabniške zgodbe morajo imeti naslednjih 6 lastnosti:

- Neodvisnost
- Prilagodljivost
- So pomembne za uporabnika oz. naročnika
- Imajo predvidljiv obseg oz. kompleksnost
- Majhnost
- Omogočajo testiranje

Bill Wake predlaga kratico INVEST (Independent, Negotiable, Valuable to users and customers, Estimatable, Small, Testable) za naštete lastnosti [14].

3.6.1 NEODVISNOST

Odvisnosti med zgodbami vodijo do problemov pri razvrščanju po prioriteti in načrtovanju iteracij. Če uporabnik določi neki zgodbi visoko prioriteto, zgodba pa je odvisna od neke druge zgodbe z nizko prioriteto, je potrebno v vsakem primeru realizirati tudi zgodbo z nižjo prioriteto. Problem se lahko pojavi tudi pri ocenjevanju časa razvoja. Če imamo 3 zgodbe, ki imajo zelo podobno funkcionalnost, bo običajno potrebno nekaj več časa za razvoj prve zgodbe, preostali dve pa bosta razviti v krajšem času, zato ne moremo oceniti kompleksnosti posamezne zgodbe, saj je ta odvisna od vrstnega reda.

Rešitev za tovrstne probleme je združevanje odvisnih zgodb v večje zgodbe ali iskanje drugačnega načina reševanja problema z drugo razdelitvijo zgodb.

3.6.2 PRILAGODLJIVOST

Uporabniške zgodbe niso pogodbe ali predpisi za programsko opremo, ampak so kratki opisi funkcionalnosti, katerih podrobnosti se bodo razreševale s pogovori med naročnikom in razvojno skupino oz. nas opominjajo, o čem je potrebno razpravljati.

Če na zgodbo gledamo kot na opomnik, je smiselno, da vsebuje naslednji dve komponenti:

- stavek ali dva, ki nas opominjata na nerešen problem,
- zapiske o postavkah, ki jih je treba razrešiti med sestankom.

3.6.3 POMEMBNOST ZA UPORABNIKA

Pri tej lastnosti je potrebno poudariti, da uporabnik in naročnik nista nujno ista oseba, vendar v tem primeru želimo, da ima zgodba čim več smisla oz. vrednosti za katerokoli izmed obeh vrst udeležencev. Primer zgodbe pri projektu z velikim številom uporabnikov (>100), ki je pomembna za naročnika in ne za uporabnika, bi bila naslednja zgodba:

- Vse konfiguracijske informacije se berejo iz centralne lokacije.

Uporabnikom je vseeno, od kod pridejo informacije, naročnik pa želi imeti na vseh enotah enako konfiguracijo. Prav tako obstajajo zgodbe, ki so relevantne za uporabnike in ne za naročnike. Obe vrsti zgodb sta sprejemljivi. Izogibati se je potrebno predvsem zgodbam, ki so pomembne samo za razvijalce in naročnikom ter uporabnikom ne pomenijo ničesar. Primer take zgodbe je naslednja zgodba:

- Vse povezave do podatkovne baze gredo preko bazena povezav (connection pool).

Kupci in naročniki takih zgodb ne morejo razvrstiti po prioriteti, zato jih je potrebno izločiti ali preoblikovati.

3.6.4 PREDVIDLJIVOST OBSEGA

Za načrtovanje je pomembno, da lahko razvijalci predvidijo obseg dela za posamezno zgodbo. Obstajajo trije razlogi, zaradi katerih to ni možno:

- Razvijalci imajo pomanjkljivo znanje o domeni zgodbe.
- Razvijalci imajo pomanjkljivo tehnično znanje.
- Uporabniška zgodba je preobsežna.

Prvi problem je rešljiv s pogovorom z naročnikom, ki je izdelal zgodbo. Drugega problema se lahko lotimo s kratkim eksperimentom, v katerem preizkusimo novo oz. neznano tehnologijo do te mere, da lahko ocenimo obseg potrebnega dela za zgodbo. Tretji problem je rešljiv z delitvijo zgodbe na zgodbe z manjšim obsegom.

3.6.5 MAJHNOST

Optimalno velikost zgodbe določajo skupina, njene zmožnosti in uporabljene tehnologije. Pri spreminjanju obsega zgodbe se poslužujemo dveh postopkov: delitev zgodb ter združevanje zgodb.

Primer zgodbe, ki je preobsežna in jo je potrebno razdeliti v manjše, je naslednja zgodba:

- Uporabnik lahko načrtuje dopust.

Včasih so lahko zgodbe premajhne, zato jih je potrebno združiti. Če imamo projekt, v katerem je potrebno odpraviti 5 napak in je za njihovo odpravljanje potreben en delovni dan, lahko vseh 5 napak združimo v eno zgodbo, saj bi izdelovanje posameznih zgodb zahtevalo preveč časa glede na opravljeno delo.

3.6.6 MOŽNOST TESTIRANJA

Zgodbe, ki jih ni možno testirati, se po navadi nanašajo na nefunkcionalne zahteve (zahteve, ki se ne nanašajo neposredno na funkcionalnost), npr.:

- Uporabnik ne sme čakati dolgo na prikaz na zaslonu.

Pri testiranju je zaželeno, da se to v čim večji meri avtomatizira. Če zgodbo iz primera preoblikujemo, jo lahko testiramo, ta test pa lahko celo avtomatiziramo in ugotovimo, ali lahko potrdimo pravilno delovanje. Zgodbo lahko preoblikujemo v naslednjo obliko:

- Prikaz na zaslonu se pojavi v času, krajšem od 2 sekund v 95% primerov.

3.7 SODELOVANJE Z UPORABNIKI

Sodelovanje z uporabniki je najpomembnejši aspekt razvoja programske opreme, vendar se mu v praksi, kot sem opazil pri lastnem delu, posveča razmeroma malo pozornosti. Razlog za to je predvsem nedostopnost pravih uporabnikov, zato se večina dogovorov sklene z različnimi posredniki, ki zastopajo uporabnike. To so lahko vodje oddelkov, upravitelji, prodajalci programske opreme, domenski strokovnjaki ter celo nekdanji uporabniki. Posredniki po navadi slabše zastopajo interese uporabnikov, saj imajo tudi lastne interese. Če se dogovarjamo na primer z vodjo izmen v klicnem centru, se bo ta bolj posvečal nadzorni funkcionalnosti, interesi podrejenih pa bodo zapostavljeni.

V nekaterih primerih imamo dostop do končnih uporabnikov, vendar je ta številčno in časovno zelo omejen. Mike Cohn v tem primeru predlaga uporabo tehnike, pri kateri zahtevamo oblikovanje uporabniške delovne skupine (user task force) [2]. Ta vsebuje toliko uporabnikov, kolikor jih je na voljo - od 2 do nekaj deset. Namen uporabniške delovne skupine je proizvajanje idej, posrednik pa še vedno ostaja tisti, ki odloča katere ideje se bodo uresničile.

Včasih pa se zgodi, da ni na voljo niti enega končnega uporabnika in se moramo zanašati na posrednika. V tem primeru lahko omilimo problem tako, da si poizkušamo izbrati več čim bolj različnih posrednikov. Poleg tega lahko poskrbimo, da čim prej pride do izdaje pripravljalne (beta) različice. Ko ta pride v roke uporabnikov, lahko ocenimo neskladja med interesom posrednika in uporabnikov.

3.8 NAJPOGOSTEJŠE NAPAKE PRI IZDELAVI ZGODB

V tem poglavju bom opisal nekatere najpogostejše napake, ki se lahko pojavijo pri razvoju z uporabniškimi zgodbami in lahko marsikaterega vodjo projekta odvrnejo od uporabe te metode za zajem uporabniških zahtev. Če se tem napakam izogibamo, lahko pohitrimo proces razvoja ter zmanjšamo količino dela, ki ni povezano neposredno z razvojem. Šele tedaj pridejo do izraza vse prednosti uporabniških zgodb.

3.8.1 PREMAJHNE ZGODBE

Majhne zgodbe pogosto povzročajo probleme z ocenjevanjem in časovnim planiranjem. To je posledica dejstva, da se ocena, ki je dodeljena majhni zgodbi, lahko popolnoma spremeni, če spremenimo vrstni red implementacije zgodb. Taka težava se pojavi v primeru naslednjih dveh zgodb:

- Rezultate iskanja lahko shranimo v datoteko XML.
- Rezultate iskanja lahko shranimo v datoteko HTML.

Ker se večji del implementacije teh dveh zgodb prekriva, je bolje, da ju združimo pred načrtovanjem (oz. sploh ne razdelimo zgodbe na dva dela, če imamo samo eno zgodbo). Ko je proces razporejanja zgodb v iteracije končan, lahko zgodbo razdelimo.

3.8.2 SOODVISNE ZGODBE

Odvisnost med posameznimi zgodbami, podobno kot premajhne zgodbe, povzroča težave pri načrtovanju. Če sta dve zgodbi soodvisni (ena ne more biti implementirana brez druge), morata biti implementirani v isti iteraciji. Ta problem se veča s številom zgodb, ki so medsebojno odvisne. Rešitev je po navadi združevanje odvisnih zgodb, v nekaterih primerih pa je potrebno ponovno razdeliti zgodbe.

3.8.3 OKRAŠEVANJE ZGODB (GOLDPLATING)

Razvijalci včasih z dobrim namenom implementirajo funkcionalnosti, ki niso bile načrtovane v iteraciji ali pa interpretirajo zgodbe preveč svobodno in presežejo zahteve, podane v zgodbah. To napako Mike Cohn poimenuje "goldplating", ki v dobesednem prevodu pomeni pozlačevanje [2].

Napaka ima psihološki izvor. Razvijalci imajo potrebo po tem, da bi prijetno presenetili uporabnike z dodatno funkcionalnostjo. Temu se lahko izognemo z doslednim vsakodnevnim sodelovanjem z naročnikom, saj razvijalci tako nimajo časa razviti nepotrebnih delov programske opreme.

Razlog lahko tiči tudi v želji razvijalcev, da bi projektu dodali "osebno noto". V tem primeru si lahko pomagamo tako, da poizkušamo izboljšati transparentnost delovnih nalog razvijalcev. V vsakem trenutku mora biti znano, kaj kdo dela. To lahko dosežemo z uvedbo kratkih dnevnih sestankov, na katerih vsakdo pove, kaj trenutno dela. Programerji lahko na ta način kontrolirajo drug drugega.

3.8.4 PREVEČ PODROBNOSTI

Ena izmed prednosti uporabniških zgodb je omejen obseg pisne dokumentacije. Večina podrobnosti je zajeta v pogovorih med naročnikom in izvajalcem. Če prezgodaj posvetimo preveč časa za določanje podrobnosti zgodbe, potem nismo izkoristili te prednosti. Tako zapravimo več časa za pisanje zgodbe, kot za pogovarjanje o njej. Tom Poppendieck [15] je duhovito opisal rešitev za ta problem - če vam zmanjka prostora za zgodbo, uporabite manjšo kartico.

3.8.5 PREZGODNJE OPISOVANJE UPORABNIŠKEGA VMESNIKA

Zgodbe, ki so napisane v začetnih stopnjah projekta (še posebej pri razvoju novih rešitev), naj ne bi vsebovale podrobnosti o uporabniškem vmesniku. Na neki točki razvoja bo sicer potrebno v zgodbe vključiti tudi podatke o uporabniškem vmesniku ali pa bodo zgodbe posredno napeljevale na določene značilnosti vmesnika, vendar se je potrebno temu izogibati, dokler je možno. Primer take zgodbe je naslednja zgodba:

- Iskalec zaposlitve si lahko ogleda informacije o podjetju na strani "Opisi delovnih mest".

Ker na začetku razvoja še ne vemo, ali bo omenjena stran sploh obstajala, bi bilo zgodbo potrebno preoblikovati:

- Ob ogledu informacij o zaposlitvi, si lahko iskalec zaposlitve ogleda informacije o podjetju.

3.8.6 Prepogosto deljenje zgodb

Lahko se zgodi, da med načrtovanjem iteracije, v želji po čim bolj optimalni razporeditvi zgodb v iteracijo, zgodbe delimo prepogosto. Zgodbo je potrebno razdeliti zaradi enega izmed dveh razlogov:

- Zgodba je prevelika, da bi bila realizirana v iteraciji.
- Zgodba vsebuje tako dele (podzgodbe) z visoko kot z nizko prioriteto, naročnik pa želi v sledeči iteraciji samo tiste z visoko prioriteto.

V obeh primerih je delitev zgodbe upravičena. Do zastajanja pride samo v primeru, ko je delitev prepogosta. Če moramo npr. deliti skoraj vse zgodbe, ki jih poizkušamo vključiti v iteracijo, je potrebno pregledati vse zgodbe in spremeniti način, na katerega smo jih napisali ali pa jih razdeliti vnaprej.

3.8.7 Težave z naročnikom

Najopaznejša težava v zvezi z naročniki je njihov odpor do pogoste interakcije in sodelovanja z razvijalci. V večjih združbah vedno obstajajo posamezniki, ki se odločajo po principu čim manjše odgovornosti. S tem se izognejo krivdi, kadar neka odločitev povzroči neuspeh, po drugi strani pa si lahko skoraj vedno lastijo vsaj del zaslug, če bi odločitev prinesla uspeh.

Če naročnik ne želi sodelovati pri določanju prioritete in razporejanju zgodb, je najbolje, da ga neformalno povabimo na zasebni razgovor, kjer poizkusimo pridobiti njegovo mnenje o pomembnih odločitvah. To moramo storiti na način, ki ne prenaša odgovornosti odločitve na naročnika. Tako bo naročnik prispeval samo svoje mnenje, odločitev in s tem odgovornost pa bo ostala na naši strani.

Poleg nesodelovanja pa se lahko zgodi, da ima naročnik težave pri določanju prioritete zgodb. V tem primeru je potrebno najprej preveriti velikost zgodb. Če so zgodbe prevelike, jih je nemogoče razporediti po prioriteti. Prav tako je nemogoče prioritizirati zgodbe, ki za naročnika nimajo posebnega pomena oz. vrednosti. Take zgodbe je potrebno preoblikovati.

3.9 Prednosti in slabosti

Najopaznejše prednosti uporabniških zgodb so:

- Spodbujanje verbalne komunikacije.
- Razumljivost (za uporabnike in razvijalce).
- Primerna velikost za načrtovanje.
- Se dobro obnesejo pri iterativnem načrtovanju.
- Spodbujajo jedrnatost opisov (brez nepotrebnih podrobnosti).
- Spodbujajo udeleženost naročnika oz. uporabnikov.

Uporabniške zgodbe pa imajo tudi nekaj slabosti. Pri velikih projektih z veliko zgodbami je težko razumeti razmerja med zgodbami. V takih primerih se včasih bolje obnesejo primeri uporabe, ki inherentno vsebujejo hierarhijo. Ta pomaga pri delu z velikim številom zahtev. Posamezen primer uporabe lahko v scenarijih predstavi zahteve, za katere bi potrebovali veliko število zgodb.

Naslednja slabost je slaba sledljivost zahtev. Če je zahtevano, da se na primer za vsako zahtevo ve, kateri test ji pripada, je potrebna dodatna dokumentacija. Rešitev tega problema sicer ni zelo zahtevna, vendar je zanjo še vedno potreben dodaten čas.

Zadnja slabost se pokaže pri zelo velikih skupinah. Pri uporabi zgodb zelo hitro raste "tiho" znanje, ki ni dokumentirano. Nekaj tega znanja je potrebno v velikih razvojnih skupinah zapisati, saj se sicer precej hitro izgubi in ne doseže vseh udeležencev.

3.10 PRIMERI

Posamezna zgodba je sestavljena iz pisnega dela, verbalne komunikacije ter testnega dela. V nadaljevanju bom predstavil izseke zgodb samo s pisnim delom, vendar je potrebno upoštevati, da zraven spadata še preostali sestavini zgodbe. Ker bom uporabniške zgodbe podrobneje obdelal še v drugem delu diplomske naloge, bom na tem mestu na kratko prikazal samo nekaj primerov.

Tipični primeri uporabniških zgodb so navedeni v naslednjem seznamu. Zgodbe se nanašajo na namišljen projekt spletne knjigarne.

- Nekatere zgodbe, ki so jih napisali kupci:
  - Uporabnik lahko dodaja knjige v košarico in jih kupi, ko zaključi z izbiranjem.
  - Uporabnik lahko odstrani knjige iz košarice pred zaključkom naročila.
  - Uporabnik lahko ocenjuje in piše kratke kritike za posamezno knjigo.
- Zgodbe, ki se nanašajo na administratorje sistema:
  - Administrator lahko dodaja nove knjige v spletno ponudbo.
  - Administrator mora odobriti ali zavrniti kritiko pred njeno objavo.
  - Administrator lahko izbriše knjigo iz spletne ponudbe.
- Zgodbe, ki predstavljajo omejitve sistema:
  - Sistem mora podpirati do 50 sočasnih uporabnikov.

4 PRIMERJAVA OPISANIH METODOLOGIJ

Ne glede na to, za katero metodo pisanja specifikacij zahtev se odločimo, je cilj čim bolje zadovoljiti potrebe naročnika oz. uporabnikov. Kot smo videli, ima vsaka metoda svoje prednosti in slabosti. Za različne vrste in velikosti projektov se te prednosti in slabosti odražajo v različni meri, pomemben faktor pa je tudi sama sestava in struktura razvojne skupine ter uveljavljen način dela v podjetju. Tako ne moremo določiti metode, ki bi bila najboljša za vse vrste projektov. Vsakokrat je pred začetkom specificiranja zahtev potrebno premisliti, kaj je najbolj pomembno. V naslednjih poglavjih bom predstavil nekatere razlike med opisanimi metodami zajema zahtev.

4.1 PRIMERI UPORABE IN UPORABNIŠKE ZGODBE

Ena največjih razlik med primeri uporabe in uporabniškimi zgodbami je njun obseg. Uporabniške zgodbe so v primerjavi s primeri uporabe krajše in enostavnejše, ker je njihova velikost omejena. Običajno za realizacijo določene uporabniške zgodbe ne predvidevamo več kot 10 dni dela razvijalcev, zato jih lahko uporabimo tudi za časovno načrtovanje razvoja. Posamezna uporabniška zgodba je ekvivalentna enemu izmed tokov dogodkov oz. scenariju v primeru uporabe, npr. osnovnemu poteku iz prejšnjega primera ustreza uporabniška zgodba "Referent lahko pošilja elektronsko pošto".

Naslednja razlika med uporabniškimi zgodbami in primeri uporabe je povezana s časom njunega obstoja. Posamezen primer uporabe obstaja v celotni fazi razvoja ter vzdrževanja projekta, medtem ko uporabniška zgodba ne preživi daljšega obdobja in je povezana z iteracijo razvoja, v kateri je bila dodana ustrezna funkcionalnost.

Zadnja razlika se nanaša na vključenost uporabniškega vmesnika v specifikacijo. Čeprav naj primer uporabe ne bi vseboval opisa uporabniškega vmesnika, se je temu včasih težko izogniti. Kot je razvidno iz prejšnjega primera, že sam osnovni tok dogodkov predpostavlja, da bo referent vpisoval naslov prejemnika ter besedilo s pomočjo tipkovnice in odposlal sporočilo s pomočjo miške. Tak opis izključuje možnost izbora prejemnika iz seznama ter vnosa besedila s prepoznavanjem govora. Uporabniška zgodba za ta primer pa je popolnoma neodvisna od uporabniškega vmesnika.

4.2 SCENARIJI IN UPORABNIŠKE ZGODBE

Glavna razlika med scenariji in uporabniškimi zgodbami je v njunem obsegu in številu podrobnosti. Scenariji vsebujejo veliko več podrobnosti in pokrivajo več uporabniških zgodb. Zgornji primer bi lahko nadomestil naslednje uporabniške zgodbe:

- Uporabnik lahko pošlje informacije o zaposlitvi prijatelju prek e-pošte.
- Uporabnik lahko ustvari življenjepis.
- Uporabnik lahko pošlje življenjepis na sedež delodajalca.
- Uporabnik lahko išče zaposlitev po geografskih regijah.

Kljub večjemu obsegu prepuščajo scenariji nekatere podrobnosti nedorečene, zato jih moramo doreči s sodelovanjem uporabnikov. Za zgornji primer bi bilo potrebno doreči naslednje:

- Se mora vsak uporabnik prijaviti v sistem z uporabniškim imenom in geslom ali je to potrebno samo za določene storitve (npr. pošiljanje e-pošte prek sistema)?
- Ali e-poštno sporočilo, ki ga dobi Jessica vsebuje vse informacije o zaposlitvi ali samo povezavo na lokacijo, kjer se nahajajo informacije?

4.3 IEEE 830 IN UPORABNIŠKE ZGODBE

Ena izmed razlik, ki sem jo opisal že v poglavju o prednostih in slabostih standarda IEEE 830 je ta, da se specifikacije po IEEE 830 za razliko od uporabniških zgodb ne osredotočajo na cilje, ki jih ima v mislih uporabnik. Poleg tega, da imajo specifikacije po IEEE 830 tudi večji obseg, pa se razlikujejo še po tem, da so cene posameznih zahtev pri IEEE 830 neznane, dokler specifikacija ni v celoti napisana.

Tipičen postopek pisanja pri IEEE 830 zahteva, da en ali več analitikov porabi dva ali tri mesece za pisanje dokumenta. Ta dokument preberejo programerji, ki ocenijo trajanje razvoja na 24 mesecev, na začetku pa je bil pričakovan razvoj v 6 mesecih. To pomeni, da tri četrtine funkcionalnosti, zapisanih v specifikacijah, ne bo realiziranih. Pri tem nismo zapravili časa samo pri pisanju odvečnih specifikacij, ampak tudi naknadno pri odločanju, katere funkcionalnosti se bodo obdržale in katere ne. Pri zgodbah je ocena zahtevnosti podana z vsako zgodbo vnaprej, zato se uporabnik lahko sam odloča na podlagi svojih želja in predvsem cene posamezne zgodbe. Ta lahko precej vpliva na to, katere funkcionalnosti bodo vključene v projekt.

5 PROJEKT EZO (EVIDENCA ZAPRTIH OSEB)

5.1 UVOD

Osnovni namen projekta EZO (Evidenca zaprtih oseb) je zagotoviti enoten pregled, vnašanje, shranjevanje in spreminjanje podatkov v zvezi z zaprtimi, pridržanimi in priprtimi osebami v zaporih na območju Republike Slovenije. Dodatna funkcionalnost sistema omogoča tudi avtomatsko generiranje dokumentov (tiralice, osebni listi, potrdila o odpustu s prestajanja kazni, ...), ki so potrebni pri pravnih in izvršilnih postopkih ter generiranje statističnih poročil za različna časovna obdobja (povprečna zasedenost zaporov, št. pobegov, št. prijetij ...).

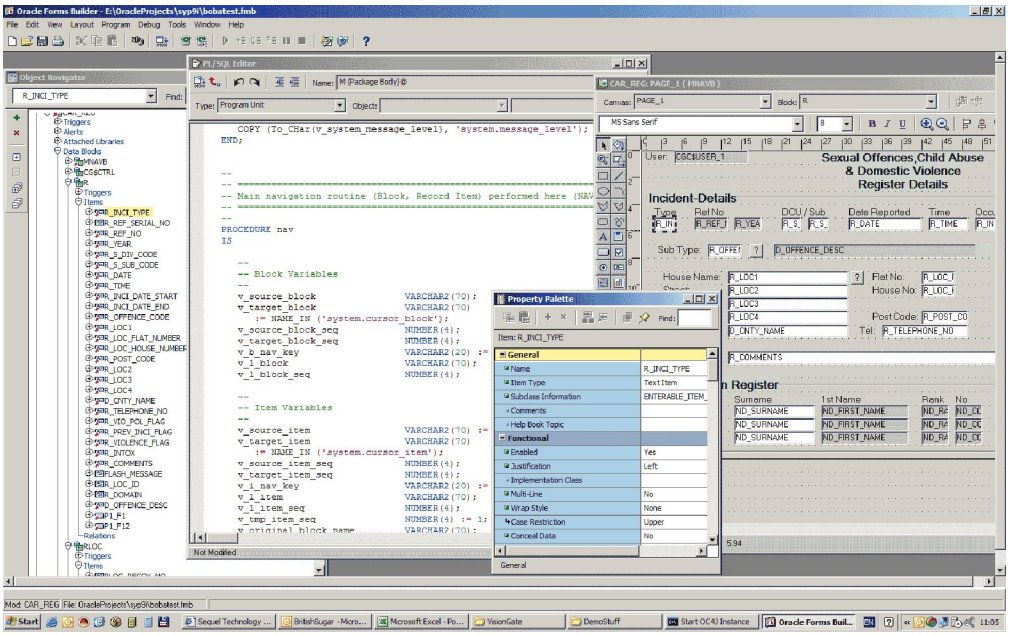
Sistem evidence zaprtih oseb, ki je trenutno v uporabi, je zaradi zastarelosti, zapletene uporabe ter nekoherentnosti in nepravilnosti v delovanju potrebno prenoviti. Naročnik nove rešitve je Uprava Republike Slovenije za izvrševanje kazenskih sankcij (URSIKS).

5.2 PREGLED UPORABLJENIH TEHNOLOGIJ

Projekt EZO je osnovan na tehnologijah podjetja Oracle Corporation, ki je vodilno na področju podatkovnih baz. Temeljne tehnologije, uporabljene v projektu so Oracle RDBMS (Oracle Relational Database Management System), ki predstavlja celotno podatkovno bazo, ter Oracle Forms, ki ga uporabljamo kot uporabniški vmesnik za interakcije med uporabniki in podatkovno bazo.

Osnovna orodja, ki smo jih uporabljali za razvoj aplikacije, obsegajo:

- Oracle Designer
- Oracle Forms
- Oracle Reports
- Oracle Discoverer



SLIKA 3: OKOLJE ORACLE FORMS DEVELOPER

5.3 ZGRADBA PROJEKTA

5.3.1 PODATKOVNI MODEL

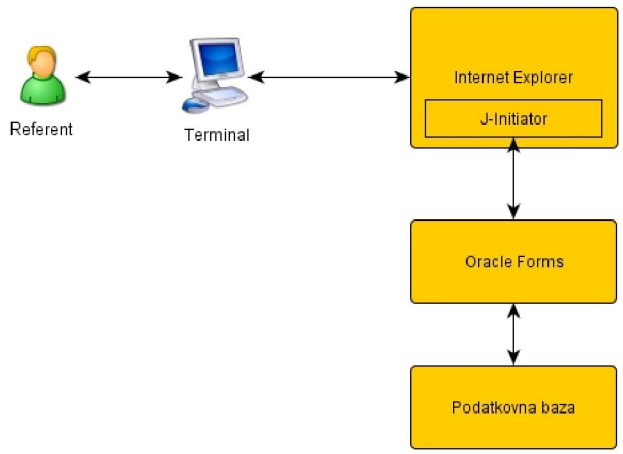
Zaradi varovanja podatkov o tehničnih podrobnostih projekta se bom omejil na okvirne podatke o shemi podatkovne baze, katerih namen je predstavitev obsega in kompleksnosti projekta. Podatkovna baza vsebuje več kot 100 tabel. Povprečno število atributov v posamezni tabeli se giblje okoli 10.

Osnovni podatki, ki se hranijo v podatkovni bazi, so podatki o kaznjencih, ki vključujejo osebne podatke, podatke o prestajanjih v določenem zavodu (zaporu), sodbah, pobegih, prekinitvah, državljanstvu itd.

Podatki, ki jih uporabnik sistema lahko izbere iz seznama vrednosti ob izpolnjevanju podatkov o obsojencu, so shranjeni v tabelah šifrantov. Te vsebujejo npr. vsa možna državljanstva, možne vrste sodb, tipe dokumentov ...

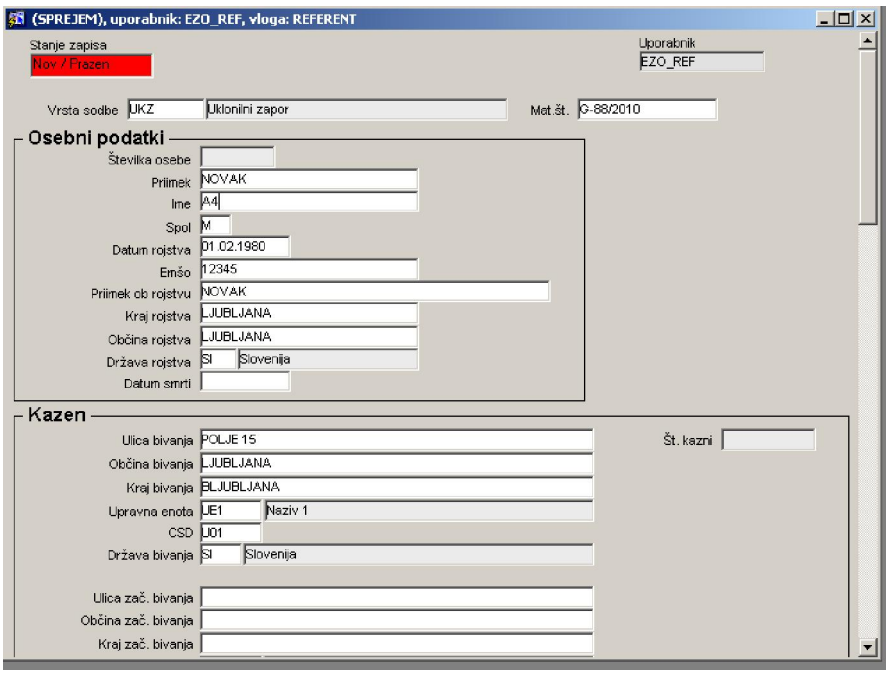
5.3.2 APLIKACIJA

Sama aplikacija, ki jo uporabljajo referenti v posameznih zavodih, teče kar v spletnem brskalniku. Na usposabljanju smo brez posebnih težav uporabljali kar Internet Explorer, ki je vgrajen v operacijski sistem Windows XP. Poenostavljen diagram delovanja aplikacije je prikazan na naslednji sliki.



SLIKA 4: POENOSTAVLJENA STRUKTURA DELOVANJA APLIKACIJE EZO

Posamezni terminali morajo imeti za delovanje aplikacije nameščen operacijski sistem z brskalnikom ter dostop do interneta. Taka zasnova omogoča spremembe v aplikaciji brez dodatnega dela na posameznih terminalih. Izgled uporabniškega vmesnika je prikazan na naslednji sliki (Slika 5).



SLIKA 5: IZGLED UPORABNIŠKEGA VMESNIKA EZO

Na sliki je prikazana zaslonska maska za vnos sprejema nove osebe. Nekatera polja so že izpolnjena.

5.4 METODA PISANJA SPECIFIKACIJ PRI PROJEKTU EZO

Med delom v podjetju RRC sem zasledil, da je posvečanje pozornosti samim specifikacijam zelo odvisno od skupine oz. oddelka, ki razvija določeno rešitev. V oddelku, v katerem sem delal, sta bila vodja projekta EZO ter vodja skupine razmeroma natančna glede pisanja specifikacij. Vsako spremembo, popravek ter tudi vsak uspešno ali neuspešno prestan test je bilo potrebno dokumentirati vsaj s krajšim opisom.

Glavni dokument, ki se nanaša na specifikacije projekta EZO, vsebuje tri poglavja. V uvodu je opisan namen dokumenta oz. zahteve. Za uvodom je podan kratek opis rešitve, v jedru dokumenta pa so navedene posamezne funkcionalne zahteve. Dokument se hierarhično deli na nivoju funkcionalnih zahtev. Vsaka zahteva vsebuje podrejeni dokument, ki predstavlja posamezno specifikacijo za določeno funkcionalnost.

Posamezne specifikacije, ki jih bom predstavil v nadaljevanju, so večinoma sestavljene iz treh poglavij. V prvem poglavju so navedene in opisane zahteve, podane s strani naročnika. Na koncu tega poglavja je navedeno, kje in kdaj je bila zahteva podana, ter pobudnik oz. pošiljatelj zahteve. V drugem poglavju je podana zasnova rešitve s krajšim pisnim sestavkom, opremljenim s slikami. Navedene so tudi omejitve in pravila za obnašanje sistema po opravljeni spremembi. Na koncu dokumenta je vselej naveden rok za izdelavo s kratkim opisom (npr.: "Funkcionalnost nameščena na testnem okolju").

5.5 TESTIRANJE

Na začetku mojega dela na projektu je bila moja glavna naloga testiranje delovanja. Pred začetkom testiranja sem porabil približno en teden, da sem se podrobneje seznanil z

delovanjem sistema ter funkcijami posameznih enot. Za projekt EZO smo uporabljali dve metodi testiranja.

Po prvi metodi sem najprej izbral določeno zaslonsko masko uporabniškega vmesnika (izdelano z Oracle Forms). Nato sem preizkušal delovanje za vse dovoljene in nedovoljene vrednosti vnosnih polj na maski. Istočasno sem spremljal spremembe v testni podatkovni bazi s pomočjo orodja Oracle Developer. Napake so se večinoma izražale kot pomanjkljiva opozorila o nepravilnem vnosu, ponekod pa je sistem tudi napačno shranjeval podatke v podatkovno bazo. Vsako od teh napak sem dokumentiral s kratkim opisom in zajemom zaslona (screenshot), kjer je bilo to smiselno.

Druga metoda je temeljila na predpripravljenih scenarijih. Najprej sva s sodelavcem pripravila okoli 60 možnih primerov uporabe sistema - scenarijev. Nekaj so jih prispevali tudi uporabniki ter naročnik. Pripravljanje scenarijev je bilo razmeroma zahtevno, saj je bilo potrebnega veliko znanja s področja poteka kazenskih postopkov. Pogosto se je bilo potrebno obračati na naročnika, ta pa ni bil vedno dosegljiv, zato je delo večkrat zastalo. Ko so bili scenariji dokončani, je bilo potrebno vsakega izmed njih izvesti na testnem sistemu. Za posamezni scenarij sem opisal izvedbo vsakega koraka in ga opremil z zajemi zaslona. Nepravilnosti v delovanju sem dokumentiral podobno kot pri prejšnji metodi.

Namen testiranja s scenariji pa ni bil samo potrditev pravilnosti delovanja. Scenarije smo po testiranju uredili v smiselne sklope ter jih združili v skupen dokument. Ta dokument smo opremili še z razlago funkcij posameznih zaslonskih mask in ga uporabili pri izobraževanju uporabnikov. Scenariji so bili seveda izbrani tako, da so uspeli zajeti večino postopkov, ki jih uslužbenci zavodov dnevno uporabljajo.

5.6 IZOBRAŽEVANJE UPORABNIKOV

Izobraževanje uporabnikov (zaposlenih v posameznih zavodih po Sloveniji) je potekalo v skupni dvorani v zavodu na Dobu. Namen izobraževanja je bil poleg učenja upravljanja z novo aplikacijo tudi sprejemanje predlogov ter kritik nastajajočega sistema.

Prvo izobraževanje se je začelo z uvodno predstavitvijo projekta ter njegove logične zasnove. Predstavili smo tudi izboljšave glede na sistem, ki je bil trenutno v uporabi. Kot je omenjeno v prejšnjem poglavju, smo na izobraževanju uporabili testne scenarije. Vsak udeleženec je prejel kopijo navodil za uporabo sistema, ki so vključevala testne scenarije (Dodatek B). Na projektorju smo prikazovali potek dela z aplikacijo, uporabniki pa so vsak na svojem računalniku izvajali operacije, ki smo jih predstavili. Obdelali smo večino testnih scenarijev ter nekaj dodatnih primerov, ki so jih predlagali referenti sami.

Težave, ki so se pojavile na izobraževanju, so bile večinoma povezane s slabim predznanjem uporabnikov o aplikaciji in na splošno delom z računalnikom, nekaj pa je bilo tudi napak v sami aplikaciji. Te napake so bile večinoma povezane s pomanjkljivim preverjanjem pravilnosti vnosa ter slabo zastavljenimi omejitvami uporabniškega vmesnika.

6 APLICIRANJE UPORABNIŠKIH ZGODB NA PROJEKT EZO

6.1 UVOD

V tem poglavju bom predstavil, kako bi izgledala specifikacija zahtev z uporabniškimi zgodbami za projekt EZO. Ob svojem delu sem imel možnost spremljati razvoj projekta EZO, zato bom lahko v nadaljevanju opisal, kako bi potekala uporaba metode uporabniških zgodb. Predpostavil bom vsaj minimalno pripravljenost naročnika za sodelovanje pri razvoju, saj sicer metode ne bi bilo mogoče izpeljati.

V poglavju "Dodatek A - Primeri specifikacij za projekt EZO" se nahajajo nekatere izmed specifikacij, ki smo jih uporabljali v podjetju RRC. Te dokumente bom pretvoril v uporabniške zgodbe, nepisno komunikacijo pa bom predstavil v besedilu. Ker so izvirne specifikacije urejene po sklopih, bo vsak sklop predstavljen v svojem poglavju. Izvirnih specifikacij nisem mogel pridobiti v celoti, zato bom nekatere uporabniške zgodbe dodal po spominu.

6.2 UPORABNIŠKE VLOGE

Vsak projekt ima določenega naročnika, ki je pobudnik za nastanek novega ali prenovo obstoječega sistema. Ker naročnik ni nujno hkrati tipični uporabnik, je potrebno določiti, na koga se bomo obrnili, ko bo potrebno oblikovati zgodbe in jih urejati po pomembnosti ter razporejati v iteracije.

Pri projektu EZO so končni uporabniki zelo slabo dosegljivi, zato se dogovarjamo s predstavnikom vseh uporabnikov, ki je tudi sam zaposlen v upravi iste ustanove in je hkrati tudi en izmed uporabnikov sistema. Ta uporabnik ima administratorske pravice, zato je potrebna posebna pazljivost, da se izognemo dajanju prednosti njegovim subjektivnim zahtevam. Poizkušati moramo zajeti zahteve večine uporabnikov.

Večinski delež uporabnikov sistema predstavljajo referenti, zaposleni v zaporih. Nekateri izmed zaposlenih imajo več pravic od ostalih. Na željo naročnika imajo možnost dostopa do sistema tudi bivši zaposleni, vendar imajo zelo omejen vpogled v podatke in so brez možnosti spreminjanja. Poleg omenjenih se lahko v sistem prijavijo tudi (maloštevilni) administratorji, ki imajo najširši nabor pravic. Možnost imajo spreminjati šifrante, popravljati podatke o zavodih, zaposlenih itd.

Ko izberemo vloge, dodamo vsaki vlogi nekaj podrobnosti in jih strnimo v seznamu oz. tabeli (Tabela 3). Pri tem sem izbral podatke o pogostosti uporabe sistema, znanje o domeni ter znanje o uporabi sistema. Te podrobnosti vloge nam med drugim pomagajo pri določanju, katere lastnosti sistema so pomembne za posamezni tip uporabnika.

Ime vloge	**Pogostost uporabe**	**Znanje o domeni**	**Znanje o uporabi sistema**
Referent	večkrat dnevno	povprečno	osnovno
Bivši zaposleni	zelo redko	povprečno	osnovno
Administrator	redko	odlično	odlično

TABELA 3: UPORABNIŠKE VLOGE

Pri tem projektu je število vlog razmeroma majhno, njihove pravice pa se deloma prekrivajo, zato same podrobnosti v zvezi z vlogo niti niso ključnega pomena. Če bi bilo število vlog večje, kot npr. v spletni trgovini, kjer bi kategorizirali kupce, bi lahko na tem koraku oblikovali tudi t. i. "persone". To so podrobno opisani namišljeni uporabniki, ki jih poimenujemo z imenom in služijo kot tipični predstavniki določene vloge.

Že od vsega začetka je smiselno med razgovorom uporabljati kartice, pri čemer vsaka kartica predstavlja posamezno vlogo. Tako lahko fizično razporedimo kartice glede na lastnosti vloge in ugotovimo, katere vloge bi bilo smiselno združiti ali odstraniti. Združujemo lahko tiste vloge, ki se ne razlikujejo dovolj, da bi bilo potrebno za njih realizirati dva različna tipa uporabnikov. Ko smo s predstavnikom uporabnikov dosegli konsenz glede števila in vrste uporabniških vlog, lahko za vsako uporabniško vlogo oz. kartico dopolnimo s kratkim opisom.

Referent	**Bivši zaposleni**	**Administrator**
Aplikacijo uporablja vsakodnevno. Potrebuje čim večjo avtomatizacijo standardnih postopkov. Nima obsežnega predznanja o uporabi računalnika ter nove aplikacije.	Aplikacijo uporablja zelo poredko. Podatkov ne spreminja. Ima samo vpogled v omejen nabor podatkov.	Aplikacijo uporablja nekajkrat na mesec. Po potrebi spreminja vsebino šifrantov in podatke o zaposlenih ter pregleduje statistične podatke.

6.3 ZGODBE

V prvem delu bom predstavil uporabniške zgodbe, ki bi jih ustvarili na samem začetku razvoja. Ker pri projektu nisem sodeloval od samega začetka, bom zgodbe oblikoval iz trenutno dostopnih podatkov ter iz lastni izkušenj z delom s sistemom. V drugem delu bom pretvoril specifikacije projekta EZO, ki so mi trenutno na voljo, v uporabniške zgodbe.

Oblikovanje uporabniških zgodb se začne s skupno delavnico, kjer razvojna skupina skupaj s predstavnikom uporabnikov posveti dve uri pisanju zgodb, ki pokrivajo čim večji del projekta. Na tem mestu se je potrebno odločiti, ali bomo zgodbe pisali naključno ("brainstorming") ali pa jih bomo postopoma obdelali vsako uporabniško vlogo. V našem primeru se odločimo za slednje. Ker se opravila, ki jih lahko izvajajo posamezne vloge deloma prekrivajo, bomo začeli z vlogo referenta, ki predstavlja večinski delež uporabnikov, nato pa dodajali zgodbe za ostale vloge.

6.3.1 ZGODBE ZA REFERENTE

Najpomembnejše uporabniške zgodbe za vlogo referenta, za katere se dogovorimo na sestanku, so naslednje:

- Referent se v sistem prijavi z uporabniškim imenom in geslom.
- Referent lahko v sistem vnese sprejem novega zapornika ali pripornika.
- Referent lahko v sistem vnese osebo, ki ni zapornik ali pripornik.
- Referent lahko pregleduje in spreminja podatke o obstoječih zapornikih.
- Referent lahko v sistem vnese odpust zapornika s prestajanja kazni.

- Referent lahko zabeleži čas in datum začetka oz. konca prekinitve prestajanja.
- Referent lahko zabeleži čas in datum dogodkov med prestajanjem (samopoškodba, zdravniški pregled, opravljanje izpita, gladovna stavka, ...).
- Referent lahko natisne tiralico, potrdilo o odpustu, osebni list ali zdravniško spričevalo.
- Referent lahko pogleda, katerim zapornikom se na določen dan izteče kazen.
- Referent lahko pregleduje fotografije zaprtih oseb.
- Referent lahko pregleduje in dodaja lažna imena ter vzdevke za zaprte osebe.
- Referent mora imeti možnost vnesti osebo brez osebnega dokumenta oz. brez podatka o imenu in priimku.
- Referent lahko pregleduje seznam oseb, ki se niso odzvale na poziv.
- Referent lahko za osebe, ki se niso odzvale na poziv, vnese ponovni poziv.

Pri večini zgodb se je potrebno s predstavnikom uporabnikov dogovoriti o podrobnostih. Pri vnosu sprejema novega zapornika in vnosu nove osebe je potrebno odgovoriti na naslednja vprašanja:

- Katere podatke se bo vnašalo v sistem?
- Kakšen bo vrstni red vnosa?
- Kakšen bo način vnosa (izbor iz seznama šifrantov, ročni vnos, potrditveno polje - "checkbox")?

Predstavnik uporabnikov na tem mestu opozori, da je pomembna lastnost tudi hitrost ter enostavnost vnosa, saj se ta izvaja zelo pogosto. Ker razvojna skupina uporablja tehnologijo Oracle Forms, se ji že na tem mestu porodi rešitev zgraditi enotno zaslonsko masko, v kateri vnašamo najpomembnejše podatke ob sprejemu nove osebe, vse pa je možno opraviti samo z uporabo tipkovnice.

Pri pregledu in spreminjanju podatkov o zapornikih je potrebno določiti, katere podatke je mogoče spreminjati. Velja npr. omejitev, da referent nima vpogleda v seznam obsojencev, ki prestajajo kazen v drugih zavodih, ampak samo v tistem, v katerem je zaposlen. O vseh podrobnostih se dogovorimo na sestanku in po potrebi kontaktiramo predstavnika uporabnikov tudi kasneje.

Eno izmed pomembnejših vprašanj je na tem mestu tudi način vnosa osebe, o kateri nimamo nobenih podatkov. Če se to dogaja pogosto, je potrebno za ta primer realizirati dodatno funkcionalnost, sicer pa lahko zaobidemo problem tako, da se namesto imena in priimka osebe uporabi podatek (šifra ali številka) na dokumentu, na podlagi katerega je bila oseba privedena v zapor oz. pripor, kasneje pa referent vnese resnične podatke o osebi, ki jo lahko poišče po številki dokumenta.

Ker je za razvoj projekta ključno tudi poznavanje tematike pravnih ter kazenskih postopkov, je komunikacija z naročnikom še toliko pomembnejša. Kot primer lahko navedem, da je potrebno pri mladoletnih osebah prekinitve prestajanja kazni upoštevati na drugačen način kot pri polnoletnih. Pri prvih se namreč določene vrste prekinitev ne prištevajo k datumu izteka

kazni. Za take vrste posebnosti ekipa razvijalcev ne more vedeti, zato mora od naročnika pridobiti čim več znanja o tematiki projekta.

6.3.2 ZGODBE ZA BIVŠE ZAPOSLENE

Bivši zaposleni imajo omejen dostop do podatkov v sistemu. Večinoma lahko samo pregledujejo podatke, ne morejo pa ničesar vnašati ali spreminjati. Ker za njih ni potrebno realizirati dodatne funkcionalnosti glede na referente, se moramo na sestanku dogovoriti o omejitvah, ki morajo veljati za bivše zaposlene. Pri tem je zopet pomembno poznavanje prava oz. zakonov o varovanju osebnih podatkov, zato ima tu naročnik ključno vlogo.

6.3.3 ZGODBE ZA ADMINISTRATORJE

Administratorji lahko izvajajo vse naloge, ki jih opravljajo referenti, poleg tega pa imajo še dodatne pravice. Ker smo zgodbe za referente že zapisali, bomo na tem mestu dodali samo dodatne zgodbe. Najpomembnejše med njimi so naslednje:

- Administrator lahko dodaja, odstranjuje ter spreminja podatke v tabelah šifrantov.
- Administrator lahko pregleduje ter spreminja podatke o zapornikih v vseh zavodih.
- Administrator lahko pregleduje in natisne statistična poročila za določena obdobja.
- Administrator lahko pregleduje in spreminja podatke o zaposlenih v vseh zavodih.

Podrobnosti, ki zanimajo razvojno skupino in jih na tem mestu prediskutiramo z naročnikom so sledeče:

- Katere vrste statističnih poročil naj bodo na voljo in kakšna je njihova struktura?
- Katere podatke o zaposlenih lahko administrator spreminja (npr. geslo in uporabniško ime morata biti skrita)?
- Katere podatke o kaznjencih lahko administrator spreminja?

6.3.4 SPLOŠNE ZGODBE

Na sestanku se moramo dogovoriti tudi o lastnostih sistema. Ko nam zmanjka idej za zgodbe s stališča uporabnikov, se posvetimo vprašanjem o potrebnih zmogljivostih in omejitvah sistema. Med najpomembnejšimi zgodbami tega dela so:

- Sistem mora omogočati sočasno uporabo vsaj 30 uporabnikom.
- Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov.
- Sistem mora skrbeti za koherentnost podatkov (Zaprta oseba ne sme prestajati kazni v dveh zavodih hkrati; Prekinitev mora biti časovno znotraj prestajanja; itd...).
- Sistem mora zaprtim osebam dodeljevati unikatne identifikacijske številke.
- Sistem mora shranjevati varnostne kopije podatkov.
- Sistem mora avtomatsko arhivirati zastarele osebne podatke zaprtih oseb ob spremembah.
- Sistem mora beležiti čas in vrsto spremembe ter uporabnika, ki je spremembo povzročil.

Pri teh zgodbah se pojavijo sledeča vprašanja, ki jih z obdelamo z naročnikom:

- Po kakšnem ključu se dodeljuje identifikacijske številke zapornikom?
- Ali so identifikacijske številke vidne uporabniku in ali jih ta lahko ročno popravlja?
- Kako pogosto se shranjuje varnostne kopije?
- Katere osebne podatke se arhivira?
- Katere spremembe se beležijo?

6.3.5 ZABELEŽKE RAZGOVOROV Z NAROČNIKOM

Med prvim sestankom, na katerem smo določali osnovne zgodbe se nam je porodilo nekaj idej, ki jih nismo zapisali v zgodbah, saj morajo same zgodbe biti enostavne ter brez nepotrebnih podrobnosti. Ker si vseh teh idej ni možno zapomniti, si nekatere izmed njih zapišemo kot dodatek posamezni zgodbi v obliki zabeležke. Njihov namen ni natančno določanje podrobnosti. Služijo samo kot opomnik, s katerim se razvijalec lažje spomni, o čem smo se na sestanku pogovarjali in do kakšnih zaključkov smo prišli. Nekaj primerov teh zabeležk za posamezne zgodbe je podanih v naslednji tabeli:

Zgodba	**Zabeležka**
"Referent lahko v sistem vnese sprejem novega zapornika ali pripornika."	"Naročnik se strinja z uporabo enotne zaslonske maske za vnos vseh podatkov."
"Referent lahko pogleda, katerim zapornikom se na določen dan izteče kazen."	"Naročnik želi, da se ob izteku na praznik ali nedeljo upošteva zadnji delovnik pred tem dnem."
"Referent lahko pregleduje seznam oseb, ki se niso odzvale na poziv."	"Naročnik želi, da se takoj po vstopu v aplikacijo izpiše opozorilo, če obstajajo osebe, ki se niso odzvale na poziv."
"Referent lahko pregleduje in spreminja podatke o obstoječih zapornikih."	"V seznamu se morajo prikazati tudi vsa lažna imena ter stara (arhivirana) imena in priimki."
"Administrator lahko pregleduje in natisne statistična poročila za določena obdobja."	"Povprečna zasedenost, št. pobegov in ostalih prekinitev prestajanja, št. predčasnih odpustov, št. sprejemov."
"Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov."	"Opozorilo naj vsebuje navedbo primera pravilnega vnosa."
"Sistem mora zaprtim osebam dodeljevati unikatne identifikacijske številke."	"Številka naj vsebuje šifro vrste sodbe ter letnico sprejema."

TABELA 4: ZABELEŽKE RAZGOVOROV Z NAROČNIKOM

Te zabeležke napišemo kar na kartice z zgodbami. S tem si tudi omejimo prostor za pisanje in tako preprečimo, da bi postale opombe preveč obsežne.

6.4 DOLOČANJE IN DELITEV PREOBSEŽNIH ZGODB

Nekatere zgodbe iz prejšnjega koraka so morda preobsežne, da bi jih lahko učinkovito razporedili v posamezne iteracije, zato jih moramo razdeliti. Po premisleku in pogovoru z naročnikom se odločimo, da bomo določene zgodbe delili na manjše. Nekaj primerov delitve zgodb na manjše je prikazanih v naslednji tabeli (Tabela 5):

Osnovna zgodba	Razdeljene zgodbe
"Referent lahko pregleduje in spreminja podatke o obstoječih zapornikih."	"Referent lahko izvede poizvedbo po imenu, priimku, lažnih imenih, vzdevkih ali identifikacijski številki zapornika."
	"Referent lahko spreminja osebne podatke zapornika."
"Sistem mora skrbeti za koherentnost podatkov (Zaprta oseba ne sme prestajati kazni v dveh zavodih hkrati; Prekinitev mora biti časovno znotraj prestajanja; itd...)."	"Sistem ne sme dovoljevati sprejema osebe, ki je že na prestajanju v drugem zavodu."
	"Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni"

TABELA 5: DELITEV UPORABNIŠKIH ZGODB

Pred ocenjevanjem zahtevnosti zgodb se s predstavnikom uporabnikov tudi dogovorimo, da bomo zgodbo "Referent mora imeti možnost vnesti osebo brez osebnega dokumenta oz. brez podatka o imenu in priimku." izločili in uporabljali že omenjeno rešitev, kjer namesto imena in priimka osebe vnesemo številko dokumenta, na podlagi katerega je bila oseba privedena.

6.5 OCENJEVANJE ZAHTEVNOSTI ZGODB

Na začetnem sestanku smo vključno z razdeljenimi zgodbami napisali 27 zgodb, ki so prikazane strnjeno v spodnji tabeli (Tabela 6). Primera razdeljenih zgodb sta označena s sivim ozadjem, primer izločene zgodbe pa s prečrtanim besedilom. Naslednji cilj je izdelava načrta izdaje (release plan), katerega namen je prikazati naročniku, kaj nameravamo realizirati v nekajmesečnem roku za izdelavo.

"Referent se v sistem prijavi z uporabniškim imenom in geslom."
"Referent lahko v sistem vnese sprejem novega zapornika ali pripornika."
"Referent lahko v sistem vnese osebo, ki ni zapornik ali pripornik."
"Referent lahko izvede poizvedbo po imenu, priimku, lažnih imenih, vzdevkih ali identifikacijski številki zapornika."
"Referent lahko spreminja osebne podatke zapornika."
"Referent lahko v sistem vnese odpust zapornika s prestajanja kazni."
"Referent lahko zabeleži čas in datum začetka oz. konca prekinitve prestajanja."
"Referent lahko zabeleži čas in datum dogodkov med prestajanjem (samopoškodba, zdravniški pregled, opravljanje izpita, gladovna stavka, ...)."
"Referent lahko natisne tiralico, potrdilo o odpustu, osebni list ali zdravniško spričevalo."
"Referent lahko pogleda, katerim zapornikom se na določen dan izteče kazen."
"Referent lahko pregleduje fotografije zaprtih oseb."
"Referent lahko pregleduje in dodaja lažna imena ter vzdevke za zaprte osebe."
~~"Referent mora imeti možnost vnesti osebo brez osebnega dokumenta oz. brez podatka o imenu in priimku."~~
"Referent lahko pregleduje seznam oseb, ki se niso odzvale na poziv."
"Referent lahko za osebe, ki se niso odzvale na poziv, vnese ponovni poziv."
"Administrator lahko dodaja, odstranjuje ter spreminja podatke v tabelah šifrantov."
"Administrator lahko pregleduje ter spreminja podatke o zapornikih v vseh zavodih."
"Administrator lahko pregleduje in natisne statistična poročila za določena obdobja."
"Administrator lahko pregleduje in spreminja podatke o zaposlenih v vseh zavodih."

"Sistem mora omogočati sočasno uporabo vsaj 30 uporabnikom."
"Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov."
"Sistem ne sme dovoljevati sprejema osebe, ki je že na prestajanju v drugem zavodu."
"Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni"
"Sistem mora zaprtim osebam dodeljevati unikatne identifikacijske številke."
"Sistem mora shranjevati varnostne kopije podatkov."
"Sistem mora avtomatsko arhivirati zastarele osebne podatke zaprtih oseb ob spremembah."
"Sistem mora beležiti čas in vrsto spremembe ter uporabnika, ki je spremembo povzročil."

TABELA 6: ZAČETNI NABOR ZGODB

Da bi lahko izdelali načrt izdaje, moramo vsaki zgodbi dodeliti določeno število točk, ki predstavljajo idealni čas, kompleksnost ali kako drugo mero, ki ima pomen za razvojno skupino. Ta postopek izvedejo razvijalci, v primeru nejasnosti pa lahko po potrebi zastavljajo vprašanja naročniku, vendar slednji ne sme (oz. ne more) sodelovati pri ocenjevanju zahtevnosti.

V našem primeru je priporočljivo, da se razvijalci pred začetkom samega ocenjevanja zahtevnosti zgodb vsaj deloma dogovorijo o strukturi podatkovnega modela, saj mora biti ta konsistenten in enostaven, kolikor je to mogoče za zahtevano funkcionalnost.

Na tem mestu bom predstavil metodo "Planning Poker", ki se uporablja pri iskanju skupne ocene zahtevnosti posameznih zgodb, kadar imamo več razvijalcev in moramo uskladiti ocene posameznih razvijalcev. Pri tej metodi najprej predstavimo zgodbo. Sledi krajši pogovor o zgodbi, nato pa vsak razvijalec postavi svojo oceno zahtevnosti, neodvisno od drugih. Nihče ne sme vedeti, kakšne ocene so si izbrali ostali, dokler ni vsak izmed razvijalcev izbral svoje ocene. To najlažje dosežemo tako, da vsak udeleženec prejme kupček praznih kartic. Razvijalec postavi oceno tako, da na eno izmed kartic zapiše svojo oceno zahtevnosti. Ocena, zapisana na kartici, ne sme biti vidna ostalim. Ko je ocenjevanje končano, razvijalci razkrijejo svoje ocene. Če se ocene razvijalcev ujemajo, je postopek ocenjevanja za to zgodbo končan. V nasprotnem primeru morata ocenjevalca, ki sta podala najvišjo ter najnižjo oceno, utemeljiti svojo odločitev, nato pa se postopek ocenjevanja izvede ponovno. Po nekaj korakih razvijalci pridejo do skupne ocene, s katero se vsi strinjajo.

V našem primeru si kot prvo zgodbo izberemo kar prvo zgodbo v seznamu: "Referent se v sistem prijavi z uporabniškim imenom in geslom.". Razvijalci se zberejo v skupnem prostoru in s seboj prinesejo večje število praznih kartic ali listov papirja. Ob prisotnosti naročnika vsak izmed razvijalcev zapiše oceno zahtevnosti za zgodbo na kartico. V našem primeru se vsi razvijalci strinjajo, da je zgodba vredna 1 točko.

Naslednja zgodba iz seznama ("Referent lahko v sistem vnese sprejem novega zapornika ali pripornika.") je nekoliko obsežnejša, poleg tega pa si jo razvijalci in naročnik lahko predstavljajo na različne načine, zato se morajo najprej pogovoriti o kompleksnosti in številu podatkov, potrebnih za vnos. Potrebovali bomo posebno zaslonsko masko za vnos. Razvijalci ter naročnik se med seboj pogovorijo o naslednjih vprašanjih, od katerih bo odvisna tudi ocena zahtevnosti zgodbe:

- Kateri podatki se bodo vpisovali na zaslonski maski?
- Kateri podatki bodo avtomatsko generirani ali izpolnjeni s privzetimi vrednostmi?
- Ali bo potrebno preverjati pravilnost vnosa podatkov (katerih)?
- Ali bo potrebno ponujati izbiro iz seznama možnih vrednosti?

Ker je zahtevnost zgodbe odvisna tudi od realizacije podatkovnega modela projekta, morajo razvijalci nekaj časa posvetiti tudi načinu shranjevanja podatkov v podatkovno bazo. Ne sme se namreč zgoditi, da bi pri realizaciji neke zgodbe začeli spreminjati podatkovni model, kasneje pa bi ugotovili, da so spremembe povzročile nepotrebne zaplete pri drugih zgodbah. Na koncu se po podobnem postopku, kot je opisan za prejšnjo zgodbo, razvijalci odločijo, da je ta zgodba vredna 4 točke.

Naslednja zgodba ("Referent lahko v sistem vnese osebo, ki ni zapornik ali pripornik.") je podobna prejšnji. Razvijalci bodo tudi tu morali realizirati zaslonsko masko za vnos, vendar pa bo ta morala vsebovati manjše število vnosnih polj in bo posledično tudi manj kompleksna. Tu se vsi strinjajo, da je zgodba krajša od prejšnje, zato ji dodelijo 3 točke.

Na enak način ocenimo tudi preostale zgodbe. Ko pridemo do zgodb, ki se nanašajo na omejitve sistema (npr.: "Sistem mora omogočati sočasno uporabo vsaj 30 uporabnikom."), jim dodelimo oceno zahtevnosti 0. Tovrstne zgodbe same ne potrebujejo časa za realizacijo, vendar vplivajo na druge zgodbe. Po končanem ocenjevanju zgodb dobimo naslednji nabor ocen:

Zgodba	**Ocena**
"Referent se v sistem prijavi z uporabniškim imenom in geslom."	1
"Referent lahko v sistem vnese sprejem novega zapornika ali pripornika."	4
"Referent lahko v sistem vnese osebo, ki ni zapornik ali pripornik."	3
"Referent lahko izvede poizvedbo po imenu, priimku, lažnih imenih, vzdevkih ali identifikacijski številki zapornika."	3
"Referent lahko spreminja osebne podatke zapornika."	3
"Referent lahko v sistem vnese odpust zapornika s prestajanja kazni."	2
"Referent lahko zabeleži čas in datum začetka oz. konca prekinitve prestajanja."	2
"Referent lahko zabeleži čas in datum dogodkov med prestajanjem (samopoškodba, zdravniški pregled, opravljanje izpita, gladovna stavka, ...)."	2
"Referent lahko natisne tiralico, potrdilo o odpustu, osebni list ali zdravniško spričevalo."	4
"Referent lahko pogleda, katerim zapornikom se na določen dan izteče kazen."	1
"Referent lahko pregleduje fotografije zaprtih oseb."	2
"Referent lahko pregleduje in dodaja lažna imena ter vzdevke za zaprte osebe."	2
"Referent lahko pregleduje seznam oseb, ki se niso odzvale na poziv."	2
"Referent lahko za osebe, ki se niso odzvale na poziv, vnese ponovni poziv."	2
"Administrator lahko dodaja, odstranjuje ter spreminja podatke v tabelah šifrantov."	4
"Administrator lahko pregleduje ter spreminja podatke o zapornikih v vseh zavodih."	2
"Administrator lahko pregleduje in natisne statistična poročila za določena obdobja."	4

"Administrator lahko pregleduje in spreminja podatke o zaposlenih v vseh zavodih."	3
"Sistem mora omogočati sočasno uporabo vsaj 30 uporabnikom."	0
"Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov."	2
"Sistem ne sme dovoljevati sprejema osebe, ki je že na prestajanju v drugem zavodu."	1
"Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni"	1
"Sistem mora zaprtim osebam dodeljevati unikatne identifikacijske številke."	2
"Sistem mora shranjevati varnostne kopije podatkov."	2
"Sistem mora avtomatsko arhivirati zastarele osebne podatke zaprtih oseb ob spremembah."	2
"Sistem mora beležiti čas in vrsto spremembe ter uporabnika, ki je spremembo povzročil."	2

TABELA 7: ZGODBE Z OCENAMI ZAHTEVNOSTI

6.6 OCENJEVANJE HITROSTI RAZVOJNE SKUPINE

Naša razvojna ekipa je vključno z menoj sestavljena iz 4 razvijalcev. Dva izmed razvijalcev imata že veliko predznanja o uporabljenih tehnologijah, saj sta sodelovala pri več projektih, kjer so uporabljali Oraclove sisteme, vendar pa imata vzporedno s tem projektom še ostale obveznosti, zato ne moreta projektu EZO posvetiti celotnega delovnega časa. Preostala dva razvijalca (jaz in sodelavec) sva na področju Oracle tehnologij še začetnika, zato bova morala nekaj časa posvetiti spoznavanju in učenju. Dogovorili smo se tudi, da bova poskrbela za dokumentacijo, uporabniška navodila, izobraževanje in pomoč uporabnikom ter sprotno testiranje sistema.

Iz povedanega sledi, da se bo naša ekipa nekoliko oddaljila od optimalne hitrosti razvoja, vendar pa bo imel naročnik toliko več časa za premislek in ugotavljanje, katere funkcionalnosti bi bilo potrebno spremeniti ali dodati.

Ker se vsi razvijalci šele spoznavamo z uporabniškimi zgodbami, poleg tega pa je projekt razmeroma unikaten, hitrosti razvojne skupine ne bomo mogli oceniti na podlagi preteklih izkušenj. Morali bomo torej ugibati na podlagi trenutno razpoložljivih podatkov.

Ko smo določali ocene zahtevnosti zgodb, smo približno definirali vrednost 1 točke kot 1 idealni dan razvoja zgodbe. V praksi moramo upoštevati veliko nepredvidljivih dejavnikov (npr. težave pri razvoju in odpravljanje nekonsistentnosti ter nesporazumov med razvijalci, sodelovanje razvijalcev pri drugih projektih, spoznavanje tehnologij), zato v oceno hitrosti vključimo nekaj rezerve. Predvidevamo, da bo razvojna skupina za 1 idealni dan razvoja realno porabila okoli 3 dni. Posamezna iteracija bo v našem primeru dolga 2 tedna oz. 10 delovnih dni. Število razvijalcev v skupini je 4, zato predvidimo za vsako iteracijo 40 programerskih dni. V 40 dneh lahko po predvidevanjih opravimo približno 14 točk, vendar si tudi tu vzamemo še nekaj rezerve in postavimo končno hitrost razvojne skupine na 12 točk. To bo največje število točk, ki jih bo pri načrtovanju vsebovala posamezna iteracija.

6.7 DOLOČANJE PRIORITET

Nalogo določanja prioritet zgodb izvede naročnik. Glavni dejavnik pri določanju prioritete zgodbe je pomembnost oz. uporabnost zgodbe za uporabnike, dodatni dejavnik pa je ocena zahtevnosti zgodbe. Občasno se zgodi, da se zgodbam s prvotno visoko prioriteto ob upoštevanju njihove zahtevnosti prioriteta zmanjša.

Na začetku postopka določanja prioritet naročnik razporedi kartice, na katerih so zapisane zgodbe, v štiri skupine glede na njihovo pomembnost realizacije v prvi izdaji. Takšen način predlaga Mike Cohn [2]. Za lažjo predstavo skupine poimenujemo z naslednjimi imeni:

- obvezno (must have)
- potrebno (should have)
- zaželeno (could have)
- ni predmet te izdaje (won't have)

Obvezne zgodbe morajo biti nujno realizirane v prvi izdaji, zato jih bomo pri načrtovanju postavili pred ostale. Za njimi pridejo potrebne zgodbe, ki sicer niso nujno potrebne za samo delovanje, vendar bi bila naša aplikacija brez njih zelo okrnjena. Za zaželene zgodbe velja, da jih bomo realizirali v prvi izdaji samo pod pogojem, da nam za njihovo realizacijo ostane dovolj časa. Ostanejo nam še zgodbe, za katere smo se že predhodno odločili da niso predmet prve izdaje.

V našem primeru nimamo zgodb, ki bi spadale v skupino " ni predmet te izdaje". Zaradi preglednosti smo jih izločili že v prejšnjih fazah načrtovanja. Ko naročnik zaključi z razporejanjem zgodb tako dobimo 3 sezname zgodb, ki so predstavljeni v naslednjih tabelah:

Zgodba	**Ocena**
"Referent se v sistem prijavi z uporabniškim imenom in geslom."	1
"Referent lahko v sistem vnese sprejem novega zapornika ali pripornika."	4
"Referent lahko v sistem vnese odpust zapornika s prestajanja kazni."	2
"Referent lahko zabeleži čas in datum začetka oz. konca prekinitve prestajanja."	2
"Referent lahko pogleda, katerim zapornikom se na določen dan izteče kazen."	1
"Referent lahko pregleduje seznam oseb, ki se niso odzvale na poziv."	2
"Referent lahko za osebe, ki se niso odzvale na poziv, vnese ponovni poziv."	2
"Administrator lahko dodaja, odstranjuje ter spreminja podatke v tabelah šifrantov."	4
"Administrator lahko pregleduje in spreminja podatke o zaposlenih v vseh zavodih."	3
"Sistem mora zaprtim osebam dodeljevati unikatne identifikacijske številke."	2

TABELA 8: OBVEZNE ZGODBE

Zgodba	**Ocena**
"Referent lahko v sistem vnese osebo, ki ni zapornik ali pripornik."	3
"Referent lahko izvede poizvedbo po imenu, priimku, lažnih imenih,	3

vzdevkih ali identifikacijski številki zapornika."	
"Referent lahko spreminja osebne podatke zapornika."	3
"Referent lahko zabeleži čas in datum dogodkov med prestajanjem (samopoškodba, zdravniški pregled, opravljanje izpita, gladovna stavka, ...)."	2
"Referent lahko natisne tiralico, potrdilo o odpustu, osebni list ali zdravniško spričevalo."	4
"Referent lahko pregleduje fotografije zaprtih oseb."	2
"Referent lahko pregleduje in dodaja lažna imena ter vzdevke za zaprte osebe."	2
"Administrator lahko pregleduje ter spreminja podatke o zapornikih v vseh zavodih."	2
"Sistem mora omogočati sočasno uporabo vsaj 30 uporabnikom."	0
"Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov."	2
"Sistem ne sme dovoljevati sprejema osebe, ki je že na prestajanju v drugem zavodu."	1
"Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni."	1

TABELA 9: POTREBNE ZGODBE

Zgodba	**Ocena**
"Administrator lahko pregleduje in natisne statistična poročila za določena obdobja."	4
"Sistem mora shranjevati varnostne kopije podatkov."	2
"Sistem mora avtomatsko arhivirati zastarele osebne podatke zaprtih oseb ob spremembah."	2
"Sistem mora beležiti čas in vrsto spremembe ter uporabnika, ki je spremembo povzročil."	2

TABELA 10: ZAŽELENE ZGODBE

6.8 NAČRT IZDAJE

Da dokončamo načrt izdaje (release plan), moramo zgodbe razporediti v iteracije. Pri načrtovanju z uporabniškimi zgodbami dopuščamo možnost, da se načrt med izvajanjem spreminja glede na hitrost razvoja. Te spremembe so v večini primerov majhne in vključujejo dodajanje ali odvzemanje posameznih zgodb iz iteracije. V našem primeru smo oceno hitrosti razvoja in zahtevnosti zgodb postavili zelo previdno, zato lahko pričakujemo, da bomo med razvojem uspeli v nekatere iteracije dodati zgodbe in realizirati celoten projekt v krajšem času. V nadaljevanju bom predstavil potek načrtovanja in prilagajanja načrta skozi celoten razvoj, za katerega se bo izkazalo, da se zaključi v 5 iteracijah.

Razvijalci predlagamo, da se v prvih dveh iteracijah realizira osnovna funkcionalnost. Ko je ta del zaključen naredimo razvojno izdajo (beta release), ker bo tako lažje oceniti, kaj manjka oz. kaj najbolj potrebujemo. S tem se bomo lahko tudi bolj približali uporabnikom, saj bodo ti imeli možnost uporabljati testni sistem ter bodo lahko po elektronski pošti ali telefonu posredovali kritike ter predloge.

Kot je opisano v poglavju 7.6, imamo v vsaki dvotedenski iteraciji na voljo 12 točk. Tako lahko zgodbe razporedimo v iteracije na način, prikazan v naslednji tabeli (Tabela 11):

1. Iteracija	Št. točk	Prioriteta
"Referent se v sistem prijavi z uporabniškim imenom in geslom."	1	obvezne zgodbe
"Referent lahko v sistem vnese sprejem novega zapornika ali pripornika."	4	obvezne zgodbe
"Referent lahko v sistem vnese odpust zapornika s prestajanja kazni."	2	obvezne zgodbe
"Referent lahko zabeleži čas in datum začetka oz. konca prekinitve prestajanja."	2	obvezne zgodbe
"Referent lahko pogleda, katerim zapornikom se na določen dan izteče kazen."	1	obvezne zgodbe
"Referent lahko pregleduje seznam oseb, ki se niso odzvale na poziv."	2	obvezne zgodbe
Vsota točk v iteraciji	**12**	
2. Iteracija	**Št. točk**	**Prioriteta**
"Referent lahko za osebe, ki se niso odzvale na poziv, vnese ponovni poziv."	2	obvezne zgodbe
"Administrator lahko dodaja, odstranjuje ter spreminja podatke v tabelah šifrantov."	4	obvezne zgodbe
"Administrator lahko pregleduje in spreminja podatke o zaposlenih v vseh zavodih."	3	obvezne zgodbe
"Sistem mora zaprtim osebam dodeljevati unikatne identifikacijske številke."	2	obvezne zgodbe
"Sistem ne sme dovoljevati sprejema osebe, ki je že na prestajanju v drugem zavodu."	1	potrebne zgodbe
Vsota točk v iteraciji	**12**	

TABELA 11: NAČRT IZDAJE

Ker nam je uspelo v dve iteraciji razporediti vse zgodbe z najvišjo prioriteto (obvezne zgodbe), na koncu dodamo še eno zgodbo iz druge skupine (potrebne zgodbe). Če po koncu 1. iteracije opazimo, da smo ocene zahtevnosti in hitrosti razvoja postavili preveč pesimistično, lahko v drugo iteracijo dodamo še dodatne zgodbe iz skupine potrebnih zgodb.

Po izvedbi prvih dveh iteracij ugotovimo, da smo hitrost razvoja ter zahtevnosti zgodb ocenili z zadovoljivo natančnostjo. V drugi iteraciji so razvijalci že vpeljani v delo na projektu in uspejo razviti celo višjo hitrost razvoja od pričakovane, zato lahko proti koncu iteracije realizirajo še dodatno zgodbo: "Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni".

Uporabniki po dveh iteracijah razvoja dobijo delujočo razvojno (beta) različico sistema. Lahko uporabljajo vse realizirane funkcionalnosti in se s tem učijo uporabe novega sistema, poleg tega pa ugotavljajo pomanjkljivosti ter podajajo predloge prek elektronske pošte ali telefona. Izkaže se, da je bila odločitev za izdajo razvojne različice aplikacije dobra, saj bi sicer med razvojem zelo težko pridobili odziv ter predloge uporabnikov, ker ti niso fizično dostopni. To je bil tudi glavni razlog za dogovarjanje s predstavnikom uporabnikov.

V naslednjih dveh iteracijah nameravamo realizirati še preostale uporabniške zgodbe. Razporedimo jih po enakem postopku kot v prejšnjem koraku. Razpored je predstavljen v naslednji tabeli (Tabela 12).

3. Iteracija	Št. točk	Prioriteta
"Referent lahko v sistem vnese osebo, ki ni zapornik ali pripornik."	3	potrebne zgodbe
"Referent lahko izvede poizvedbo po imenu, priimku, lažnih imenih, vzdevkih ali identifikacijski številki zapornika."	3	potrebne zgodbe
"Referent lahko spreminja osebne podatke zapornika."	3	potrebne zgodbe
"Referent lahko zabeleži čas in datum dogodkov med prestajanjem (samopoškodba, zdravniški pregled, opravljanje izpita, gladovna stavka, ...)."	2	potrebne zgodbe
"Sistem mora omogočati sočasno uporabo vsaj 30 uporabnikom."	0	potrebne zgodbe
"Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni."	1	potrebne zgodbe
Vsota točk v iteraciji	**12**	
4. Iteracija	**Št. točk**	**Prioriteta**
"Referent lahko natisne tiralico, potrdilo o odpustu, osebni list ali zdravniško spričevalo."	4	potrebne zgodbe
"Referent lahko pregleduje fotografije zaprtih oseb."	2	potrebne zgodbe
"Referent lahko pregleduje in dodaja lažna imena ter vzdevke za zaprte osebe."	2	potrebne zgodbe
"Administrator lahko pregleduje ter spreminja podatke o zapornikih v vseh zavodih."	2	potrebne zgodbe
"Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov."	2	potrebne zgodbe
Vsota točk v iteraciji	**12**	

TABELA 12: NAČRT IZDAJE (2. DEL)

Uspelo nam je razporediti vse zgodbe iz skupin "obvezne zgodbe" ter "potrebne zgodbe". Po zaključku četrte iteracije se lahko posvetimo še zgodbam iz skupine "zaželjene zgodbe". Te lahko v celoti realiziramo v eni sami iteraciji, saj je vsota njihovih zahtevnosti enaka 10 točk, hitrost razvojne skupine pa je 12 točk na iteracijo. Ker pa nimamo dovolj časa, da bi jih vključili v prvo izdajo, bodo te zgodbe postale predmet druge izdaje.

6.9 SPREJEMNI TESTI

Sprejemne teste po navadi začnemo pisati že pred samim razvojem, vendar se dodajajo ter spreminjajo skozi celoten razvoj, zato jih bom predstavil šele v sledečem besedilu.

Sprejemni testi omogočajo, da določimo, ali je zgodba razvita do te mere, da lahko naročnik sprejme ta del programske opreme. To pomeni, da je za določanje sprejemnih testov odgovoren predvsem naročnik, vendar pa skoraj vedno potrebuje pomoč nekoga iz razvojne skupine. Po navadi imamo na projektu v razvojni skupini nekoga, ki je odgovoren za testiranje na novo razvitih funkcionalnosti. V našem primeru je razvojna skupina tako majhna, da se ta oseba posveča testiranju samo del časa. Večino testiranja opravljava jaz in sodelavec

in zato tudi komunicirava z naročnikom z namenom dobre definicije sprejemnih testov. V nadaljnjem besedilu bom opisal nekaj primerov sprejemnih testov za določene uporabniške zgodbe.

Začnimo z naslednjo zgodbo:

- Referent lahko v sistem vnese sprejem novega zapornika ali pripornika.

Sprejemni testi za to zgodbo morajo zaobjemati čim več različnih tipov sodb in dokumentov, na podlagi katerih je bila oseba privedena. Prav tako moramo testirati preverjanje pravilnosti vnosa z vnašanjem različnih vrednosti v polja za vnos datuma in časa. Ta del se sicer v osnovi nanaša na zgodbo "Sistem mora opozoriti uporabnika pri vnosu napačnih podatkov", vendar bomo pri vsaki zgodbi zaradi večje zanesljivosti opravili tovrstne teste sproti, na koncu pa bomo teste opravili ponovno za celoten sistem. Skupaj z naročnikom določimo naslednje sprejemne teste:

- Uporabi vse možne vrste sodb in dokumentov, ki so na izbiro.
- Poizkusi vnesti neveljaven vnos za vsako polje na maski za sprejem.
- Poizkusi vnesti osebo, ki je trenutno že na prestajanju kazni.
- Poizkusi vnesti osebo, ki je že bila na prestajanju kazni, a je bila iz zapora že odpuščena v preteklosti.
- Poizkusi vnesti nič, eno ali več državljanstev.

Naslednja zgodba, za katero bom opisal sprejemne teste, se nanaša na poizvedbo po podatkih o zaprti osebi. Pri poizvedbah je potrebno testirati vsa polja ter čim več njihovih kombinacij, po katerih je možno izvesti poizvedbo ter preveriti ali je sistem res vrnil prave podatke. Pravilnost vrnjenih podatkov je seveda odvisna tudi od pravilnosti shranjevanja podatkov ob vnosu, zato bomo s temi testi ugotavljali tudi napake pri shranjevanju.

Pri tovrstnem testiranju imamo na voljo dve vrsti testnih podatkov. Prvi so podatki, s katerimi napolnimo podatkovno bazo takoj po postavitvi z uporabo skripte, ki generira in v bazo vpiše določeno število namišljenih zaprtih oseb ter ostalih podatkov. Druga vrsta podatkov pa je rezultat vnosa preko uporabniškega vmesnika. Prve bom v nadaljevanju imenoval "generirani zapisi", druge pa "vneseni zapisi". Poizvedbe najprej testiramo na vnesenih zapisih in če pride do napake ponovimo test na generiranih zapisih. Tako lahko na enostaven način ugotovimo, ali se napaka pojavlja pri poizvedbi ali pri vnosu. Kot primer si oglejmo naslednjo zgodbo:

- Referent lahko izvede poizvedbo po imenu, priimku, lažnih imenih, vzdevkih ali identifikacijski številki zapornika.

Za to zgodbo postavimo naslednje sprejemne teste:

- Uporabi vrednosti ter njihove kombinacije, ki ustrezajo vsaj enemu uporabniškemu zapisu v podatkovni bazi.
- Uporabi vrednosti ter njihove kombinacije, ki ne ustrezajo nobenemu zapisu v podatkovni bazi.
- Poizkusi vnesti neveljavno identifikacijsko številko zapornika.

- Uporabi vrednosti ter njihove kombinacije, ki ustrezajo vsaj enemu generiranemu zapisu v podatkovni bazi.

Oglejmo si še zgodbo za administratorje:

- Administrator lahko pregleduje in natisne statistična poročila za določena obdobja.

Pri tej zgodbi je bilo pred določitvijo testa potrebno natančno določiti vrste in strukturo poročil ter obliko izpisa. Sprejemni testi za to zgodbo so sledeči:

- Poizkusi natisniti vse vrste statističnih poročil za obdobja v preteklosti s koncem obdobja nastavljenim na datum, ki je manjši ali enak trenutnemu.
- Poizkusi natisniti vse vrste statističnih poročil za obdobja v prihodnosti.
- Poizkusi uporabiti neveljaven vnos v poljih za datum začetka in konca obdobja.

Kot zadnji primer poglejmo še naslednjo splošno zgodbo, ki predstavlja lastnost sistema:

- Sistem ne sme dovoljevati vnosa prekinitve prestajanja zunaj časovnih okvirov prestajanja kazni.

Pri tej zgodbi se dogovorimo za naslednje sprejemne teste:

- Poizkusi vnesti prekinitev prestajanja znotraj časovnih okvirov prestajanja.
- Poizkusi vnesti prekinitev prestajanja zunaj časovnih okvirov prestajanja.
- Poizkusi vnesti prekinitev prestajanja znotraj časovnih okvirov prestajanja v preteklosti za zapornika, ki je bil na prestajanju v preteklosti in je trenutno na prostosti.
- Poizkusi vnesti prekinitev prestajanja zunaj časovnih okvirov prestajanja v preteklosti za zapornika, ki je bil na prestajanju v preteklosti in je trenutno na prostosti.

Pri tej zgodbi se pojavi tudi vprašanje, ali je mogoč vnos prekinitve prestajanja za pretekla prestajanja. Želja naročnika je, da se vnos prekinitve v preteklosti onemogoči v primeru, da je bila oseba pred tem že odpuščena iz zapora, saj je vnos prekinitve smiseln samo za trenutna prestajanja.

6.10 Prevedba obstoječih specifikacij

V tem poglavju bom pretvoril konkretne specifikacije, ki smo jih uporabljali pri razvoju projekta EZO v uporabniške zgodbe. Namen poglavja je predstaviti enostavnost uporabniških zgodb ter njihovo zmožnost, da zaobjamejo vse, kar v zvezi s specificiranjem zahtev potrebujemo v običajnem postopku razvoja.

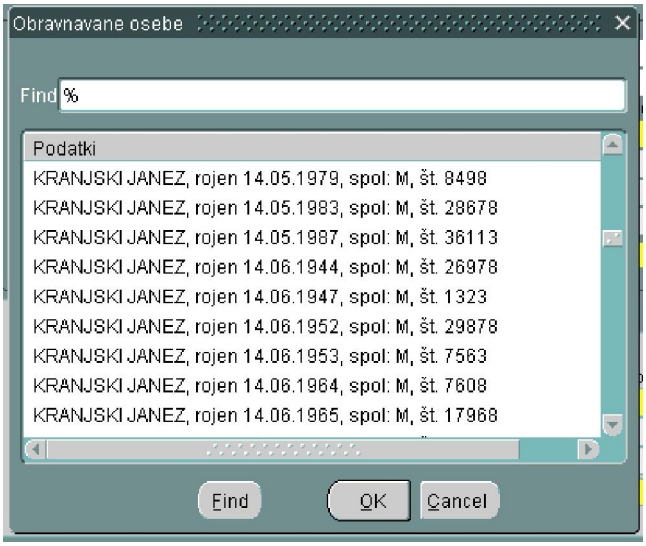
Izvorne specifikacije se nahajajo v prilogi (str. 54) in se nanašajo na vzdrževanje in nadgradnjo sistema. Če nadaljujemo razvoj projekta, kot sem ga opisal v prejšnjih poglavjih, lahko te zgodbe vključimo v že napisan nabor zgodb in jih realiziramo v naslednjih iteracijah, dokler ne pride do naslednje izdaje aplikacije.

6.10.1 ARHIVIRANJE OSEB

Specifikacija o arhiviranju oseb se nanaša na spremembo sistema, s katero omogočimo shranjevanje starih podatkov o osebi, kadar se ti podatki spremenijo. Ideja se je porodila s strani naročnika, v našem primeru je to predstavnik projektne skupine e-pravosodje. Uporabniška zgodba za to funkcionalnost bi izgledala takole:

- Uporabniku se pri sprejemu nove osebe / zapornika ponudijo tudi podatki iz arhiva starih podatkov o osebah.

Ko referent vnaša podatke ob sprejemu osebe, se zaradi lažjega in hitrejšega vnosa prikaže seznam obstoječih oseb, ki so bile kdajkoli vnesene v sistem. Opisani popravek v tem seznamu prikaže tudi stare podatke, ki so bili naknadno popravljeni. Za lažjo predstavo je na naslednji sliki (Slika 6) prikazan seznam, iz katerega lahko referent izbere obstoječo osebo.



SLIKA 6: IZBOR OBRAVNAVANE OSEBE

Podatkovna baza je v trenutku nastanka slike napolnjena s testnimi podatki. Ker smo podatke o osebi "Janez Kranjski" spreminjali, so se ti arhivirali, ob izboru osebe pa imamo na voljo tudi stare podatke.

6.10.2 DRŽAVLJANSTVO

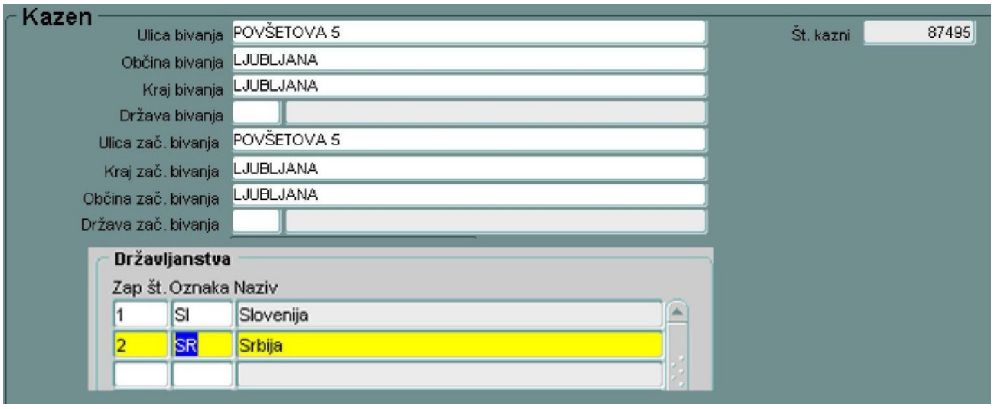
Na enotni zaslonski maski za sprejem kaznovane osebe sprva ni bil zahtevan vnos podatka o državljanstvu. Naročnik si je naknadno zaželel tudi to funkcionalnost, zato smo napisali sledečo zgodbo:

- Referent lahko ob sprejemu osebe na enotni maski vnese državljanstvo kaznovane osebe.

Ker ima določena oseba lahko več kot eno državljanstvo, smo se s predstavnikom uporabnikov dogovorili, da lahko na enotni maski vnesemo do vključno tri državljanstva. Če ima oseba več kot tri državljanstva, jih mora referent vnesti na posebni zaslonski maski za vnos in spreminjanje podatkov o kazni za zaprto osebo. Ta dogovor zapišemo tudi kot zabeležko:

- Vnos 3 državljanstev na enotni maski. Sicer se vnašajo na maski za vnos kazni.

S tem smo zabeležili dodatno funkcionalnost. Zgodbo lahko dodamo v nabor zgodb. Po realizaciji je spremenjen tudi izgled zaslonske maske za vnos sprejema nove osebe (Slika 7).



SLIKA 7: VNOS DRŽAVLJANSTVA NA MASKI ZA SPREJEM

6.10.3 VNOS POZIVA

Na enotni maski za hiter vnos si je naročnik zaželel tudi vnos podatkov o pozivih. Na tem mestu je tudi izrazil željo, da se vnos poziva umesti med sklopa "kazen" in "dokument" na tej maski. Podatki o pozivu morajo vključevati podatke o sodišču, opravilni številki, datumu poziva, predvidenem nastopu, razlogu odloga, zavodu, času, kraju ter organu prijetja. Zapišemo naslednjo zgodbo:

- Referent lahko ob sprejemu osebe na enotni maski vnese podatke o pozivu.

Na kartico zgodbe je potrebno dodati še zabeležko o podrobnostih vnosa:

- Sodišče, opravilna št., datum, predviden nastop, razlog odloga, zavod, čas, kraj, organ prijetja.

Zgodbo ponovno dodamo v nabor zgodb.

6.10.4 UPRAVNA ENOTA

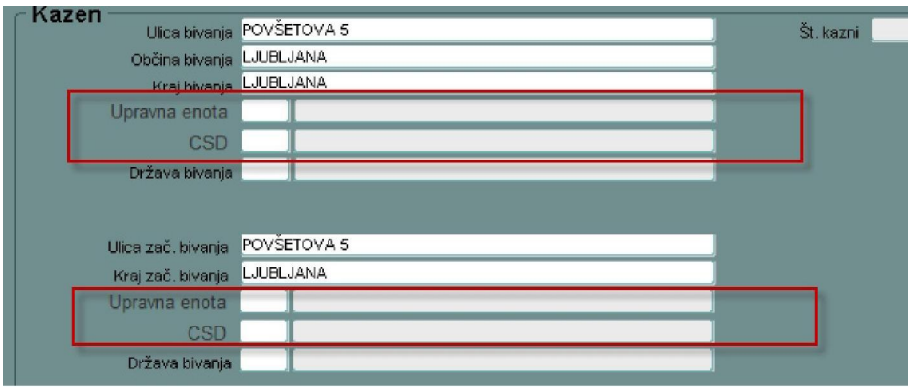
S strani naročnika je bil na enotni maski zahtevan vnos podatkov o centru za socialno delo in upravni enoti stalnega in začasnega bivališča kaznovane osebe. Zgodba, ki smo jo napisali na sestanku izgleda takole:

- Referent lahko na enotni maski vnese podatke o centru za socialno delo in upravni enoti.

Poleg zgodbe na kartico dodamo še sledečo zabeležko:

- Vnos upravne enote ločen za stalno ter začasno bivališče.

S tem bodo razvijalci imeli dovolj podatkov za razvoj funkcionalnosti. Nova verzija zaslonske maske za vnos sprejema bo izgledala, kot je prikazano na naslednji sliki (Slika 8).



SLIKA 8: VNOS UPRAVNE ENOTE IN CSD

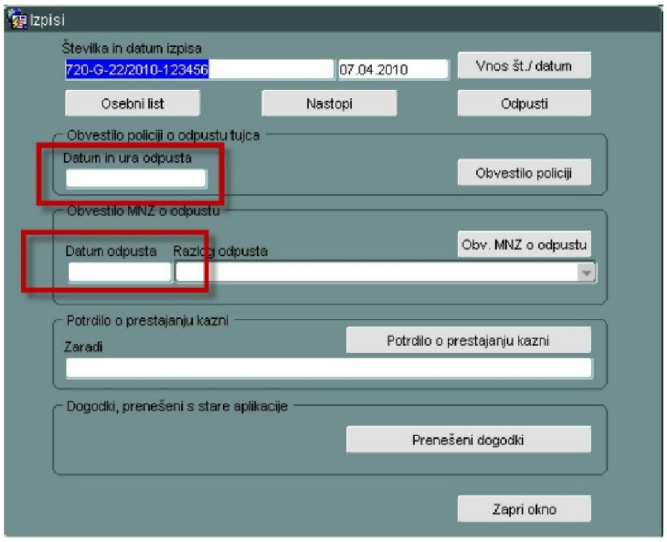
Z rdečo obrobo so označena dodana polja.

6.10.5 NOVI POPRAVKI

Na dveh dodatnih sestankih z naročnikom smo se dogovorili še za nekatere dodatne spremembe v aplikaciji. Najprej smo določili, da mora biti na maski za klicanje izpisov (potrdila, osebni list, obvestila) že vpisan privzeti datum odpusta iz prestajanja kazni za določeno kaznovano osebo. S tem smo olajšali delo referentu, saj mu datuma ni treba vpisovati ročno. Zgodba izgleda takole:

- Referentu se na maski za izpise v poljih "datum in ura odpusta" ter "datum odpusta" avtomatsko prikaže privzeta vrednost.

Zaslonska maska, na katero se ta zgodba nanaša, je prikazana na naslednji sliki (Slika 9).



SLIKA 9: ZASLONSKA MASKA ZA KLICANJE IZPISOV

Ker je predstavnik uporabnikov ugotovil, da se izpisi pogosto uporabljajo kar pri sprejemu nove osebe, bomo postopek olajšali s tem, da bomo dodali na dno enotne maske za sprejem gumb, s katerim se odpre okno za klicanje izpisov:

- Referent lahko iz enotne maske prikliče masko za klicanje izpisov.

Pod zabeležke dodamo še naslednji stavek:

- Na dno maske se doda gumb "Izpisi".

Naslednja sprememba, ki jo je naročnik predlagal se nanaša na okence, kjer se izpišejo podrobnosti izračuna kazni. Trenutno se izpišeta samo podatka "ID kazni" ter "ID dokumenta". Želeli bi, da se poleg tega izpiše še opravilna številka dokumenta, zato tvorimo zgodbo:

- Uporabnik si lahko v okencu podrobnosti izračuna izteka kazni ogleda tudi opravilno številko dokumenta.

Dogovorili smo se tudi glede oblike izpisa, zato dodamo zabeležko:

- Opravilna št.: <številka>

Opazili smo tudi nepotrebno funkcionalnost na zaslonski maski za pregled kazni. Ko uporabnik izbere obravnavano osebo, je potrebno klikniti na gumb "Prikaži", da se prikažejo podatki o prestajanjih. Ker se vse skupaj dogaja v istem oknu, ni razloga, da bi uporabnik moral uporabiti gumb, saj je podatki lahko izpišejo avtomatsko ob izboru osebe. Tu dodamo zgodbo:

- Podatki o prestajanjih se v maski za pregled kazni prikažejo avtomatsko ob izboru osebe.

Pod zabeležke za vsak primer dodamo še naslednji stavek:

- Gumb "Prikaži" ni več potreben in se izloči.

V isti zaslonski maski je naročnik opazil, da se v polju, kjer je izpisan dokument, podatki ne prikažejo v celoti, ker je polje premajhno in je besedilo odrezano. Tu z naročnikom pregledamo možne rešitve. Ena izmed možnosti bi bila povečanje polja in s tem tudi celotne maske, vendar se v tem primeru še vedno lahko zgodi, da bodo podatki preobsežni, poleg tega pa je velikost okna omejena tudi z resolucijo zaslona, ki v večini primerov ni zadostna. Na koncu se odločimo za rešitev, kjer se lahko uporabnik pomakne v polje s pomočjo bližnjice na tipkovnici ter pregleda vsebino polja s pomikanjem levo in desno. Dodamo zgodbo:

- Uporabnik se lahko pomakne v polje "dokument" v maski za pregled kazni z uporabo bližnjice, po polju pa se premika z uporabo tipkovnice.

Nazadnje je naročnik izrazil dve dodatni želji. Prva je sprememba imena aplikacije iz "Evidenca zaprtih oseb" v "Intranetna evidenca kaznovanih oseb". Druga zahteva pa je povečanje velikosti delovne površine. Ta velikost je trenutno manjša od osnovnega okna vmesnika, generiranega z Oracle Forms, želeli pa bi, da se razteza preko celotnega okna. Na tem mestu dodamo dve novi zgodbi:

- Ime aplikacije naj bo Intranetna evidenca kaznovanih oseb.
- Višina in širina delovne površine naj se prilegata notranjemu robu okna aplikacije.

S tem smo zaključili sestanek o predlaganih spremembah s strani naročnika.

6.10.6 MANJŠE SPREMEMBE

Na enem izmed sestankov z naročnikom so se ponovno pojavile nove zahteve po spremembah. Naročnik je pripravil naslednji seznam zahtev:

- Sprememba matične številke osebe ob avtomatičnem generiranju v primeru mladoletnih obsojencev (uporabi se oznaka MO, pri sodnem pridržanju pa SP).
- Na enotni maski za sprejem mora biti pri vnosu sodišča poziva dovoljen samo vnos okrožnih sodišč.
- Ob vnosu pripora na enotni maski za sprejem naj se v primeru čakajoče kazni ali v primeru, da je oseba na prekinitvi, prikaže obvestilo.
- Na enotni maski za sprejem naj se odstrani potrditveni polji (checkbox) za čakajočo in umaknjeno kazen.
- Na enotni maski za sprejem naj bo dovoljen le vnos sodbe, ki lahko nosi prestajanja.

Po pogovoru o podrobnostih tvorimo naslednje zgodbe:

- Matična številka za obsojence mora vsebovati oznako MO, za sodno pridržanje pa SP.
- Pri vnosu sodišča poziva je uporabniku dovoljen samo vnos okrožnih sodišč.
- Če ima oseba čakajočo kazen ali je na prekinitvi, mora uporabnik ob vnosu na enotni maski za sprejem prejeti obvestilo.
- Polji za čakajočo in umaknjeno kazen na enotni maski se odstranita.
- Uporabnik ima na enotni maski za sprejem možnost vnašati le sodbe, ki lahko nosijo prestajanja.

6.10.7 KONTROLA PRVEGA ZNAKA

Ker razvojna skupina v zasnovi ni upoštevala, da uporabniki niso vešči dela s sistemom, se je porodila zahteva po kontroli prvega znaka vnosa. Kontrola se izvrši ob vnosu imena, priimka in opravilne številke. Če se zgodi, da je prvi znak vnosa številka ali presledek, mora biti uporabnik na to opozorjen, sistem pa vnosa ne shrani in zahteva popravek. Dodamo zgodbo:

- Uporabnik mora biti opozorjen ob nepravilnem vnosu imena, priimka ali opravilne številke sodbe.

Po zabeležke pa dodamo:

- Prvi znak ne sme biti presledek ali številka.

6.11 PRIMERJAVA OBEH METOD

V tem podpoglavju bom na kratko primerjal interno metodo pisanja specifikacij v podjetju RRC z metodo uporabniških zgodb in predstavil njune prednosti ter slabosti. S prevedbo specifikacij v prejšnjem poglavju (6.10) sem pokazal, da je možno na enostaven način obstoječe specifikacije prevesti v obliko uporabniških zgodb.

Metodi se razlikujeta predvsem v podrobnosti opisa posameznih funkcionalnosti. Interna metoda podjetja RRC predpisuje standardno obliko dokumentov. Ti so urejeni hierarhično,

kot je razvidno v poglavju Dodatek A (str. 54). Na vsakem sestanku z naročnikom se zabeleži vse zahteve, te pa se naknadno združijo v en dokument, ki predstavlja posamezno naročilo ter vsebuje specifikacije zahtev, za katere smo se dogovorili na enem ali nekaj zaporednih sestankih.

Slabost opisane metode je zahtevnost pisanja samega dokumenta za specifikacijo. Ker je izdelovanje specifikacije prezahtevno, ga ne moremo izvesti neposredno z naročnikom na sestanku. To pomeni, da naročnik dokument vidi šele kasneje, običajno nekaj dni po sestanku, poleg tega pa ne sodeluje neposredno pri njegovem nastajanju in ga mora pred začetkom razvoja pregledati ter potrditi, da se strinja z napisanim. Če razvijamo aplikacijo, pri kateri se med razvojem zahteve naročnika velikokrat spremenijo, nam tak način dela otežuje razvoj. Zabeležiti moramo namreč vsako spremembo oz. dodatno funkcionalnost v specifikacijah zahtev. Tu se pokaže prednost metode uporabniških zgodb.

Metoda uporabniških zgodb je zasnovana na način, ki dovoljuje hitro prilagajanje spremembam zahtev in zahteva le minimalen čas za izdelavo pisne specifikacije (spomnimo se, da pisni del uporabniške zgodbe obsega le stavek ali dva). Z uporabo te metode nam po samem sestanku z naročnikom ni potrebno dodatno delo v zvezi s specifikacijami, poleg tega pa naročnik ni primoran po nekaj dneh ponovno pregledati in potrditi specifikacije, saj pri njenem nastajanju sodeluje neposredno na sestanku.

Na prvi pogled se z uporabniškimi zgodbami znebimo odvečnega dela in s tem prihranimo čas za izdelavo specifikacije, vendar pa je potrebno poudariti, da se nekaj tega prihranka izniči zaradi potrebe po pogostejši in obsežnejši komunikaciji med naročnikom in razvijalci. Na sestanku pri izdelavi uporabniških zgodb moramo opraviti več dela, kot bi bilo potrebno pri običajem dogovarjanju (po interni metodi RRC) o zahtevah. Prav tako se nam lahko zgodi, da razvoj zaradi nenatančnosti specifikacije poteka počasneje in je večkrat prekinjen, ker morajo razvijalci ugotoviti, kaj določena zgodba dejansko zahteva (to je seveda odvisno od tega, kako dobro so napisane zgodbe, vendar se problemu ne moremo popolnoma izogniti).

Druga pomanjkljivost uporabniških zgodb v primerjavi z metodo podjetja RRC pa se pokaže pri razreševanju sporov z naročnikom v zvezi z zahtevano in realizirano funkcionalnostjo. Naročnik mora biti pripravljen sodelovati in biti seznanjen s postopki ter načinom razvoja z uporabniškimi zgodbami. Ker si naročnika (ali njegovega predstavnika) po navadi ne moremo izbrati sami, nam lahko njegova nepripravljenost na sodelovanje povzroča težave. Predvsem bi poudaril, da to ni slabost same metode. Potrebno je pretehtati, katera metoda je primerna v primeru naročnika, ki ne sodeluje. Kadar imamo težave pri dokazovanju, kaj je bilo zahtevano s strani naročnika in kaj ne, se torej interna metoda podjetja RRC izkaže za boljšo, saj je vsaka zahteva opisana do podrobnosti. Če imamo v rokah dokument, ki ga je naročnik potrdil, se lahko nanj sklicujemo, medtem ko se na uporabniško zgodbo, ki obsega tudi ustno komunikacijo, ne moremo sklicevati.

7 ZAKLJUČEK

V začetnih poglavjih sem predstavil nekaj metod pisanja specifikacij programske opreme in prikazal njihove prednosti ter slabosti. V osrednjem delu diplomske naloge sem podrobneje opisal metodo uporabniških zgodb ter nato primerjal vsako izmed opisanih metod z metodo uporabniških zgodb.

V prejšnjem poglavju mi je uspelo simulirati razvoj programske rešitve z uporabniškimi zgodbami kot metodo zajema zahtev. Pri tem sem se omejil na najpomembnejše dele razvoja in poizkušal izkoristiti vse prednosti uporabniških zgodb, ki bi lahko prišle do izraza na tem projektu.

Primerjava metode uporabniških zgodb z metodo podjetja RRC, ki smo jo uporabljali v praksi, je smiselna zaradi več možnih izboljšav razvoja. Z opisom poteka razvoja z uporabniškimi zgodbami sem nakazal možne spremembe v pristopu k zajemu zahtev, ki bi izboljšale učinkovitost in kakovost programske opreme.

Slabosti trenutnega načina so predvsem posledica slabe agilnosti ter pomanjkljivega sodelovanja z naročnikom ter uporabniki. S tega gledišča prinašajo uporabniške zgodbe veliko koristi. Sodelovanje je implicitno vgrajeno v definicijo uporabniških zgodb, prav tako pa tudi spremembe ne zahtevajo veliko časa in ne zapletajo razvoja po nepotrebnem.

V nekaterih pogledih se trenutni način zajema zahtev izkaže za boljšega. Ker je v našem primeru naročnik državni organ, se je v praksi izkazalo, da vsi udeleženci nimajo popolnoma enakih interesov, nekateri pa so bili celo proti postavitvi novega sistema. Ko je bilo treba predstaviti delovanje ter prednosti glede na stari sistem, je včasih prišlo tudi do konfliktov o tem, kaj je bilo dogovorjeno ter čigava je krivda za določene pomanjkljivosti. Pri tem se izkaže, da se najbolje obnese natančna specifikacija zahtev v obliki pisnega dokumenta, v katerem so navedene vse podrobnosti, prav tako pa vsebuje tudi ime pobudnika zahteve. Tako se lahko razvijalci zaščitijo v primeru nesporazumov, saj se lahko sklicujejo na točno določen dokument. Če ta določene postavke ne vsebuje, to pomeni, da ni bila zahtevana.

Rezultat tega diplomskega dela je torej ugotovitev, da je metoda uporabniških zgodb primerna za uporabo na večjih projektih in je, kljub pomanjkljivostim, še vedno zelo uporabna, saj omogoča hiter razvoj ter zanesljivo rešitev.

8 DODATEK A - PRIMERI SPECIFIKACIJ ZA PROJEKT EZO

8.1 GLAVNI DOKUMENT

Specifikacija zahtev

02.02.003
Vzdrževanje in nadgradnja EZO
Naročilo št. 3

Vsebina

Zgodovina sprememb

Verzija	Datum	Avtor	Opis spremembe

Uvod

1.1 Namen

Naročnik (Uprava RS za izvrševanje kazenskih sankcij (URSIKS)) ima potrebo po dopolnitvi in nadgraditvi informacijskega sistema EZO in IEZO po pogodbi.

Opis rešitve

1.2 Produkt

Zahteve projekta se nanašajo na obstoječi IS Evidenca zaprtih oseb (EZO) in na novi IEZO. Z implementacijo rešitve bo prišlo do nadgradnje obeh sistemov.

1.3 Namestitev

Rešitev bo nameščena kot del posodobitve produktov EZO in iEZO.

Funkcionalne zahteve

1.4 EZO-10-01: Popravek – Arhiviranje oseb

Zahteva se shranjevanje vseh podatkov tabele obsojenci, v kolikor se vsebina spremeni. V aplikaciji se pri izbiri osebe na enotni maski ponudijo tudi podatki iz arhivirane tabele.


SP-Arhiviranje%20oseb.doc

1.5 EZO-10-02: Popravek – Kontrola prvega znaka

Zahtevana je kontrola na vnosu imena, priimka in opravilne številko sodbe, ki preverja, če je prvi vneseni znak presledek ali številka. V kolikor se to zgodi, potem mora biti uporabnik opozorjen.


SP-Kontrola%20prvega%20znaka%20v2.

1.6 EZO-10-03: Popravek – Poziv na enotni maski

Zahtevan je vnos podatkov o pozivih na maski za sprejem osebe. Podatke o pozivih se umesti med sklopa »kazen« in »dokument«, potrebno pa je dodati podatke o sodišču(šifrant), opravilni številki, datumu poziva, predvidenemu nastopu, razlogu odloga(šifrant) in zavodu(šifrant).

Poleg podatkov o pozivih se doda še čas prijetja, kraj in organ(šifrant) iz maske o prijetjih.

Pri vnosu novih podatkov na maski za sprejem osebe se upoštevajo vse omejitve in poslovna pravila, ki so že implementirana na maskah za pozive in prijetja.


SP-Poziv%20na%20enotni%20maski.doc

1.7 EZO-10-04: Popravek – Upravna enota in CSD

Zahtevan je vnos podatkov o centru za socialno delo in upravni enoti bivališča kaznovane osebe kot tudi začasnega bivališča kaznovane osebe. Podatki se polnijo v šifrantu iz katerega lahko uporabnik pri vnosu podatkov izbira vrednosti.


SP-Upravna%20enota%20in%20CSD.doc

1.8 EZO-10-05: Popravek – Vnos državljanstva

Zahtevan je vnos državljanstva tudi na enotni masko za sprejem kaznovane osebe. Uporabnik mora imeti možnost vnesti tri državljanstva hkrati. V kolikor ima kaznovana oseba več kot tri državljanstva jih uporabnik vnese na maski za vnos kazni.


SP-Vnos%20državljanstva.doc

1.9 EZO-10-06: Popravki aplikacije

Različni popravki aplikacije, ki so bili ugotovljeni kot potrebni med testiranjem aplikacije.


SP-Manjše%20spremembe.doc


SP%20-%20Novi%20popravki.doc

1.10 EZO-10-07: Priprava poročil

Popravek poročil v iEZO glede na seznam usklajenih predlog, ki so bile posredovane s strani projektne skupine

1.11 EZO-10-08 Priprava testnih scenarijev

Priprava testnega scenarija, ki je namenjen kot pripomoček za testiranje aplikacije

1.12 EZO-10-09 Izvedba izobraževanja uporabnikov

Priprava in izvedba sedmih izobraževanj za uporabnike aplikacije iEZO

- Prvo izobraževanje (20.4. in 22.4.)
- Izobraževanje projektne skupine (6.5.)
- Drugo izobraževanje (11.5. in 13.5.)
- Predstavitev izpisov (15.5. in 17.5)

1.13 EZO-10-10 Sestanki

Izvajalec se udeležuje rednih in izrednih delovnih sestankov, ki so nujno potrebni za usklajevanje nalog in izpolnjevanje terminskega načrta

1.14 EZO-10-11 Prehod v produkcijo

Aktivnosti v povezavi s prehodom na produkcijo:

- Priprava scenarija za prehod
- Popravki migracijske procedure
- Spremljanje prehoda
- Testiranje migracije podatkov

1.15 EZO-10-12: Koordinacija na projektu

Za izpolnitev vseh elementov specifikacij »02.02.003« mora izvajalec koordinirati izvajanje posamezni nalog med naročnikom, izvajalcem, 3Gen in MJU.

1.16 EZO-10-13: Testiranje aplikacije

Testiranje delovanja aplikacije s strani izvajalca.

1.17 EZO-10-14: Umik modula za vnos uporabnikov

Umik vseh funkcionalnosti v zvezi s kreiranjem, upravljanjem uporabnikov in njihovih pravic znotraj aplikacije.

1.18 EZO-10-15: Podpora uporabnikom

Nudenje podpore uporabnikom v času testiranja in prehoda v produkcijo. Med aktivnosti se šteje beleženje telefonskih klicev in sporočil na elektronski pošti, iskanje in reševanje težav, pomoč uporabnikom in zagotovljena razpoložljivost.

8.2 POSAMEZNE SPECIFIKACIJE

8.2.1 ARHIVIRANJE OSEB

SPECIFIKACIJA

Arhiviranje oseb

V S E B I N A

ZGODOVINA DOPOLNITEV DOKUMENTA

Ime dokumenta	Datum zadnje spremembe	Avtor spremembe	Opis spremembe
SP-Arhiviranje oseb.doc	10.03.2010	Rok Berdajs	Nov dokument.

1 ZAHTEVE

Zahteva se shranjevanje vseh podatkov tabele obsojenci, v kolikor se vsebina spremeni. V aplikaciji se pri izbiri osebe na enotni maski ponudijo tudi podatki iz arhivirane tabele.

1.1 REFERENCE

Pošiljatelj - pobudnik	Projektna skupina e-pravosodje
Datum	09.03.2010
Zadeva	Sestanek – pregled analize razlik in potrditev popravkov

2 ZASNOVA REŠITVE

2.1 SPREMEMBE PODATKOVNEGA MODELA

Podatkovni model se nadgradi z novo tabelo: arhiv_obsojenci, kamor se prepiše podatek o obsojencu, v kolikor se v izvorni tabeli zanj spremenijo podatki.

Nova tabela arhiv_obsojencev vsebuje spodnje atribute:

Atribut	Tip	opis
obs_id	number(15)	Id obsojenca in povezava na tavelo obsojenci
priimek	varchar2(50)	Priimek
ime	varchar2(50)	Ime
spol	varchar2(1)	Oznaka spola
emso	varchar2(13)	emšo
datum_rojstva	date	Datum rojstva
datum_smrti	date	Datum smrti
priimek_ob_rojstvu	varchar2(80)	Priimek ob rojstvu
kraj_rojstva	varchar2(60)	Kraj rojstva
obcina_rojstva	varchar2(60)	Občina rojstva
dre_id_rojstva	varchar2(3)	Enolična oznaka države
um	varchar2(30)	Uporabniško ime, ki je shranilo spremembo
dm	date	Datum spremembe

2.2 SPREMEMBA ISKALNIKA IMEN

Pri vnosu v enotni maski se s premikom na polje priimek prikaže sezanm vseh oseb. Zahteva se prikaz vseh priimkov in imenov – tudi arhivskih. Vrne se vedno obsojenec_id iz tabele obsojenci.

2.3 PRIKAZ ARHIVSKIH PODATKOV

Za pregledovanje arhivske tabele obsojencev se pripravi z orodjem Oracle Discoverer pregled vseh arhiviranih podatkov za izbrano osebo. Osebo se bo iskalo po arhiviranem imenu in priimeku.

Rok za izdelavo:

01.04.2010 – Funkcionalnost nameščena na testnem okolju

8.2.2 DRŽAVLJANSTVO

SPECIFIKACIJA

Vnos državljanstva

V S E B I N A

ZGODOVINA DOPOLNITEV DOKUMENTA

Ime dokumenta	Datum zadnje spremembe	Avtor spremembe	Opis spremembe
SP-Vnos državljanstva.doc	10.03.2010	Aleš Kravos	Nov dokument.

1 ZAHTEVE

Zahtevan je vnos državljanstva tudi na enotni masko za sprejem kaznovane osebe. Uporabnik mora imeti možnost vnesti tri državljanstva hkrati. V kolikor ima kaznovana oseba več kot tri državljanstva jih uporabnik vnese na maski za vnos kazni.

1.1 REFERENCE

Pošiljatelj - pobudnik	Projektna skupina e-pravosodje
Datum	09.03.2010
Zadeva	Sestanek – pregled analize razlik in potrditev popravkov

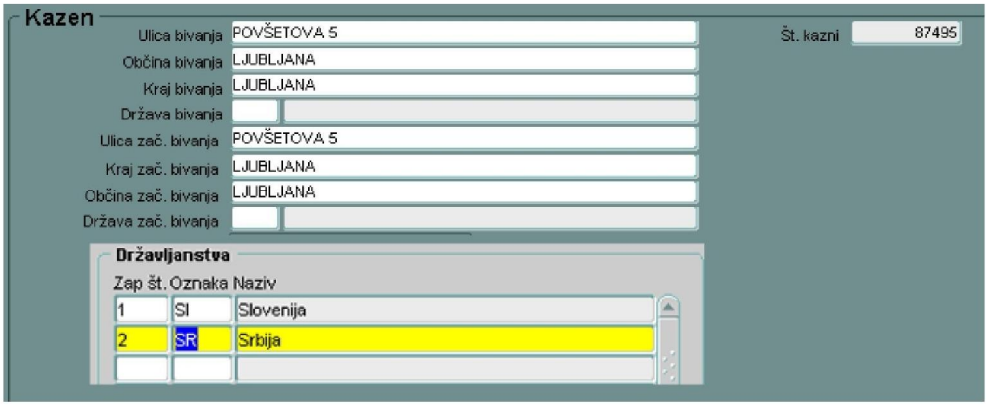
2 ZASNOVA REŠITVE

2.1 NOVA POLJA

Doda se nova polja na enotni maski za sprejem kaznovane osebe in na maski za vnos kazni:

Maska: SPREJEM (enotna maska za sprejem kaznovane osebe)
Pot do maske: Pregledi >> Seznam oseb v zavodu >> Gumb Sprejem nove osebe

Omogoči se vnos državljanstva, ki se vnaša za podatki o bivališču in začasnem bivališču. Uporabnik lahko vnese največ tri različna državljanstva.



2.2 PRAVILA

Zaporedna številka se določi samodejno, tako da se pri prvem zapisu ponudi številka »1«, na drugem številka »2« in na tretjem številka »3«. Uporabnik lahko spremeni zaporedje s spreminjanjem teh številk.

Uporabnik se najprej s tabulatorjem v prvo vrstico premakne iz polja »Država zač. bivanja«. Na to se pomakne v drugo in tretjo vrstico in na koncu v polje »Zaposlen pred nastopom« oz. naslednje polje na enotni maski v primeru, da se bo razporeditev polj na maski spremenila.

Rok za izdelavo:

01.04.2010 – Funkcionalnost nameščena na testnem okolju

8.2.3 VNOS POZIVA

SPECIFIKACIJA

Vnos poziva na maski za sprejem osebe

V S E B I N A

ZGODOVINA DOPOLNITEV DOKUMENTA

Ime dokumenta	Datum zadnje spremembe	Avtor spremembe	Opis spremembe
SP-Poziv na enotni maski.doc	10.03.2010	Grega Rožac	Nov dokument.

1 ZAHTEVE

Zahtevan je vnos podatkov o pozivih na maski za sprejem osebe. Podatke o pozivih se umesti med sklopa »kazen« in »dokument«, potrebno pa je dodati podatke o sodišču(šifrant), opravilni številki, datumu poziva, predvidenemu nastopu, razlogu odloga(šifrant) in zavodu(šifrant).
Poleg podatkov o pozivih se doda še čas prijetja, kraj in organ(šifrant) iz maske o prijetjih.
Pri vnosu novih podatkov na maski za sprejem osebe se upoštevajo vse omejitve in poslovna pravila, ki so že implementirana na maskah za pozive in prijetja.

1.1 REFERENCE

Pošiljatelj - pobudnik	Projektna skupina e-pravosodje
Datum	09.03.2010
Zadeva	Sestanek – pregled analize razlik in potrditev popravkov

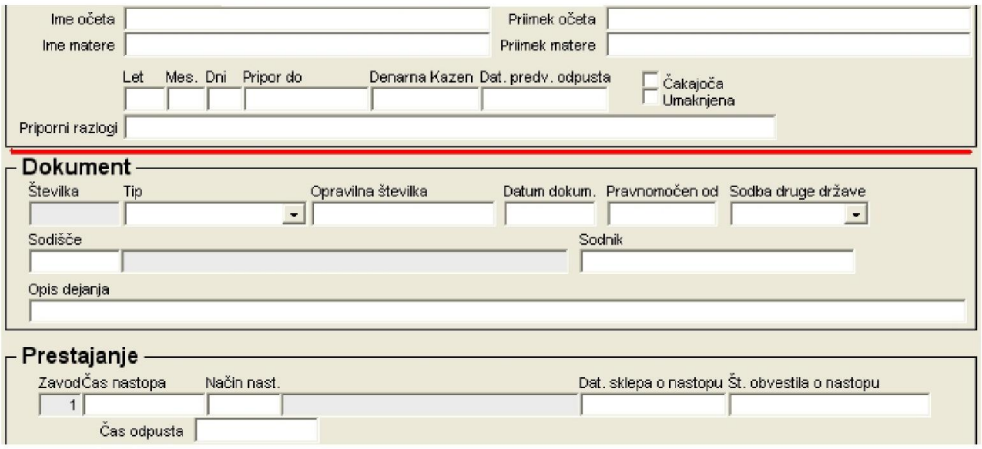
2 ZASNOVA REŠITVE

2.1 NOVA POLJA

Doda se nova polja na enotno masko za sprejem kaznovane osebe:

Maska: SPREJEM (enotna maska za sprejem kaznovane osebe)
Pot do maske: Pregledi >> Seznam oseb v zavodu >> Gumb Sprejem nove osebe

Omogoči se vnos podatkov o pozivih, ki se se umestijo med sklopa »kazen« in »dokument«. Dodajo se podatki o sodišču(šifrant), opravilni številki, datumu poziva, predvidenemu nastopu, razlogu odloga(šifrant) in zavodu(šifrant).
Poleg podatkov o pozivih se doda še čas prijetja, kraj in organ(šifrant) iz maske o prijetjih.
Pri vnosu novih podatkov na enotni maski se upoštevajo vse omejitve in poslovna pravila, ki so že implementirana na maskah za pozive in prijetja.



2.2 PRAVILA

Pri vnosu podatkov o sodišču, opravilni številki, datumu poziva in zavodu se preverja obveznost polj. Pri poljih o sodišču, razlogu odloga, zavodu in organu je ob vnosu možnost izbire podatka iz ustreznega šifranta, preveri se pravilnost vnosa, prav tako pa se na maski avtomatično prikažejo tudi nazivi iz šifranta.

Rok za izdelavo:

01.04.2010 – Funkcionalnost nameščena na testnem okolju

8.2.4 UPRAVNA ENOTA

SPECIFIKACIJA

Vnos podatka o upravni enoti in centru za socialno delo

V S E B I N A

ZGODOVINA DOPOLNITEV DOKUMENTA

Ime dokumenta	Datum zadnje spremembe	Avtor spremembe	Opis spremembe
SP-Upravna enota in CSD.doc	10.03.2010	Aleš Kravos	Nov dokument.

1 ZAHTEVE

Zahtevan je vnos podatkov o centru za socialno delo in upravni enoti bivališča kaznovane osebe kot tudi začasnega bivališča kaznovane osebe. Podatki se polnijo v šifrantu iz katerega lahko uporabnik pri vnosu podatkov izbira vrednosti.

1.1 REFERENCE

Pošiljatelj - pobudnik	Projektna skupina e-pravosodje
Datum	09.03.2010
Zadeva	Sestanek – pregled analize razlik in potrditev popravkov

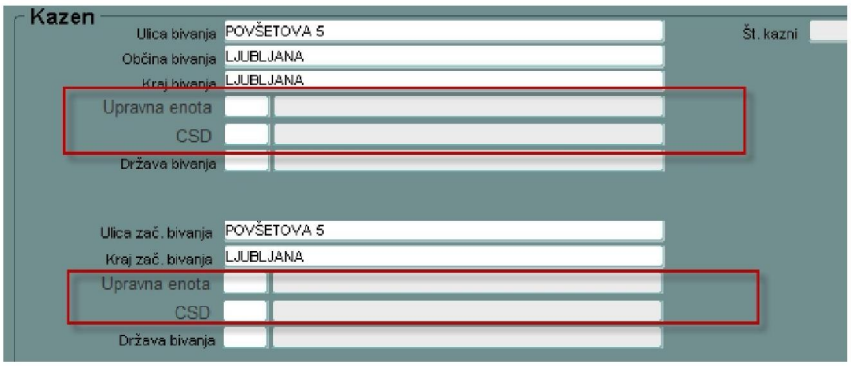
2 ZASNOVA REŠITVE

2.1 NOVA POLJA

Doda se nova polja na enotni maski za sprejem kaznovane osebe in na maski za vnos kazni:

Maska: SPREJEM (enotna maska za sprejem kaznovane osebe)
Pot do maske: Pregledi >> Seznam oseb v zavodu >> Gumb Sprejem nove osebe

Omogoči se vnos podatka kot izbira iz šifranta za podatkom o kraju bivanja in pred državo bivanja.

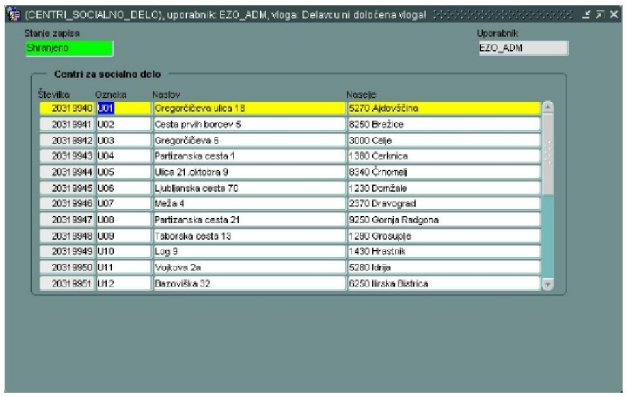


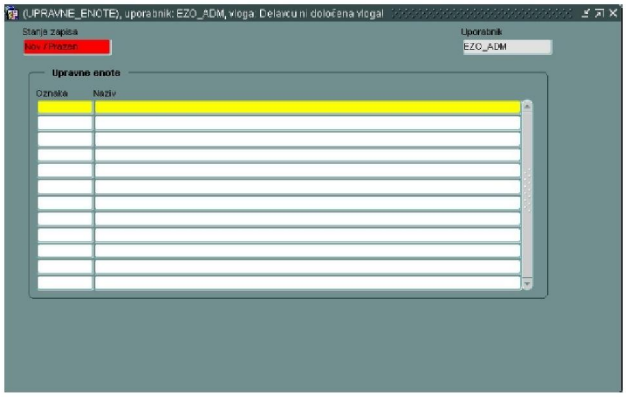
Maska: KAZEN (maska za vnos kazni)
Pot do maske: Pregledi >> Seznam oseb v zavodu >> Gumb Kazni >> Zavihek identiteta

Omogoči se vnos podatkov na enak način kot na enotni maski za sprejem kaznovane osebe.

2.2 UREJANJE ŠIFRANTOV

Vrednosti se vnašajo v obstoječih šifrantih za centre za socialno delo in upravne enote, ki jih lahko urejajo samo administratorji aplikacije:





3 OBVEZNOSTI NAROČNIKA

Naročnik mora izvajalcu posredovati šifrant centrov za socialno delo in seznam upravnih enot. Seznam se pošlje v .xls datoteki.
Datoteka za polnjenje CSD mora vsebovati spodnje atribute:
Oznaka
Naslov
Naselje

Datoteka za polnjenje UE mora vsebovati spodnje atribute:
Oznaka
Naziv

Rok za izdelavo:

01.04.2010 – Funkcionalnost nameščena na testnem okolju

8.2.5 NOVI POPRAVKI

RRC Računalniške storitve, d.d., Jadranska 21, p.p. 3407, SI-1001 Ljubljana, Slovenija
T +386 1 477 85 24 F +386 1 425 52 29 S www.rrc.si E info@rrc.si

UPORABNIŠKA SPECIFIKACIJE

NOVI POPRAVKI

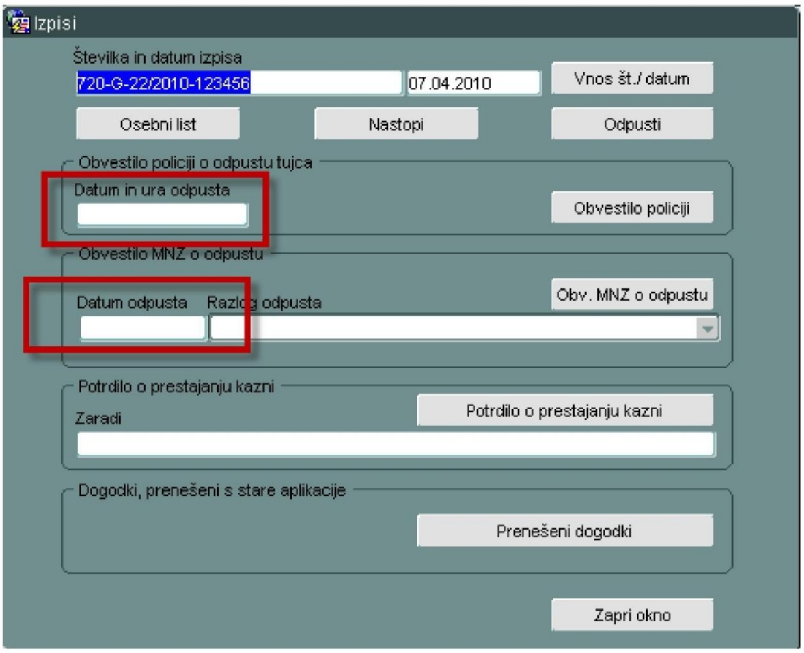
Datum: 10.03.2010
Pripravil: Aleš Kravos
Reference:

- Sestanek 02.04.2010 (Dvornik, Kravos, Berdajs)
- Sestanek 06.04.2010 (Dvornik, Kravos)

1. PRIVZET DATUM ODPUSTA

Zaželeno je, da na maski za klicanje izpisov privzame vrednost zadnjega datuma odpusta na osnovi tega iz kje se maska kliče. V kolikor se kliče maska iz prestajanja, potem naj privzame datum odpusta iz prestajanja, če se kliče iz kazni potem pa datum zadnjega odpusta.

Maska za klicanje izpisov:



Maska se trenutno kliče iz maske za vnos kazni (KAZNI) in maske za prestanja (PRESTAJANJA).

Kasneje še iz maske SPREJEM (glej točko 2.).

2. KLICANJE IZPISOV NA MASKI ZA SPREJEM

Na enotni maski za sprejem kaznovane osebe (SPREJEM) je zaželeno, da se na dnu maske doda gumb »Izpisi«, ki kliče masko za izpis osnovnih izpisov.

Na masko naj se prenesejo vse spremenljivke, ki se že prenašajo, če se maska kliče iz maske KAZNI in PRESTAJANJA.

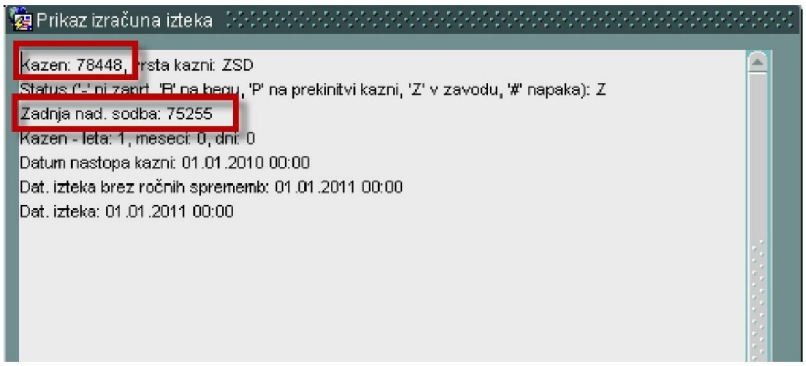
3. IZPIS OPRAVILNE ŠTEVILKE PRI PODROBNOSTIH IZRAČUNA KAZNI

Pri izpisu podrobnosti izračuna kazni se trenutno izpiše samo ID kazni in ID dokumenta.

V obeh primerih je zaželeno, da se poleg izpiše še opravilna številka, ki je v tabeli DOKUMENTI v polju OPRAVILNA_STEVILKA. Tabela KAZNOVANE_OSEBE se na tabelo DOKUMENTI poveže preko tujega ključa DOK_ID.

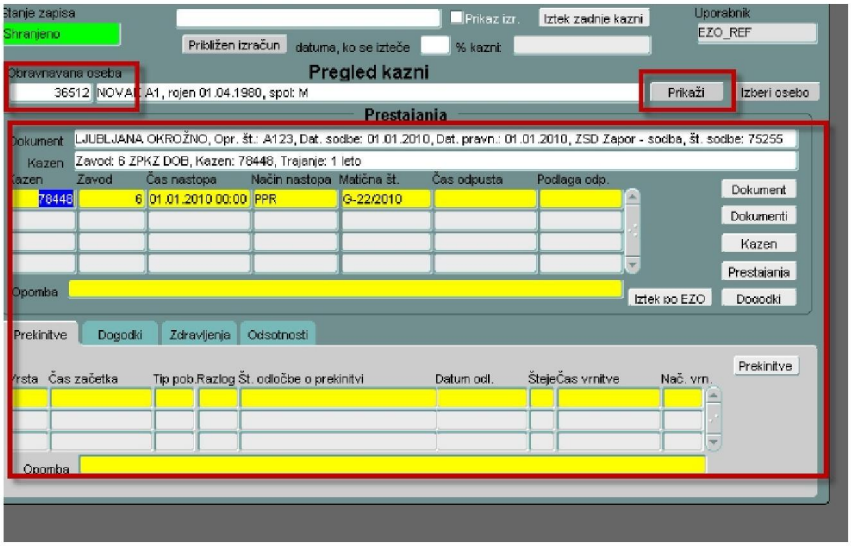
Izpiše naj se pri vsaki navedbi kazni ali dokumenta v obliki »Opravilna št.: <številka> «

Torej se mora izpisati tudi pri izpisu vključenih kaznih.



4. TAKOJŠNIH PRIKAZ PODATKOV ZA KAZNOVANO OSEBO

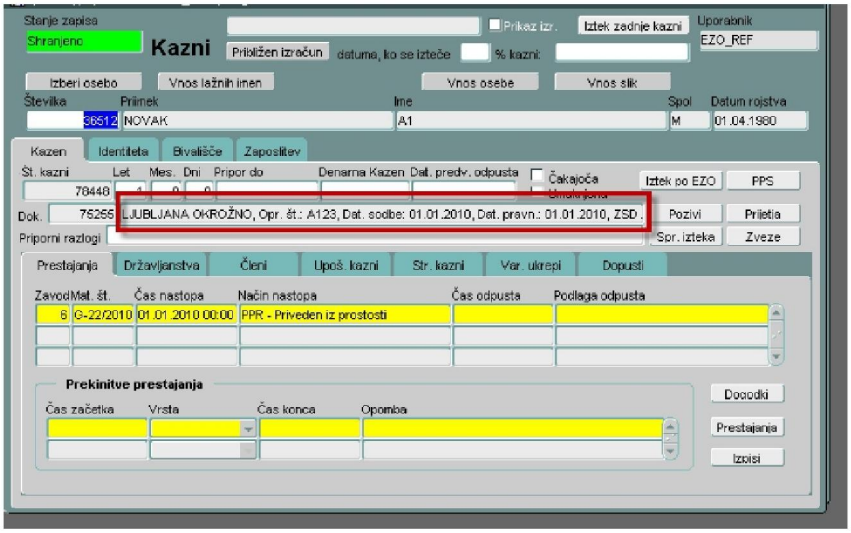
Želeni je, da se na maski KAZNI_PREGLED podatki v bloku »Prestajanja« prikažejo takoj, ko se izbere oseba v polju »Obravnavana oseba« in ne šele po tem, ko se klikne gumb »Prikaži«, tako kot to deluje sedaj.



5. PREGLED PODATKOV V POLJU DOKUMENT

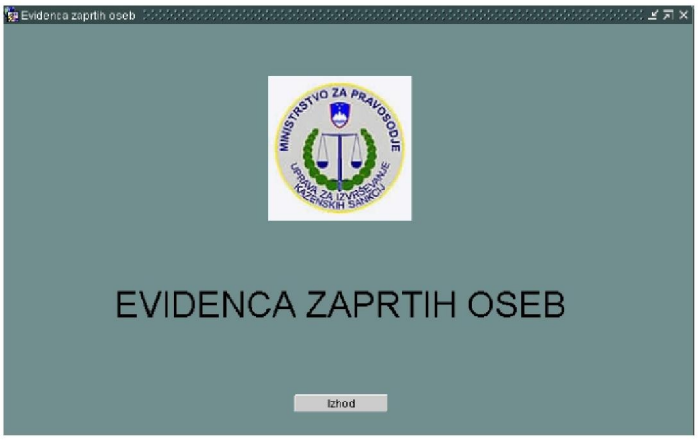
Na maski kazen se podatke v polju dokument težko pregleduje, saj je polje premajhno in je besedilo odrezano.

Potrebno je razviti rešitev, kot je na primer uporaba funkcije CTRL+E, ki pa trenutno ni možno saj se s cursorjem ne moremo postaviti v polje.



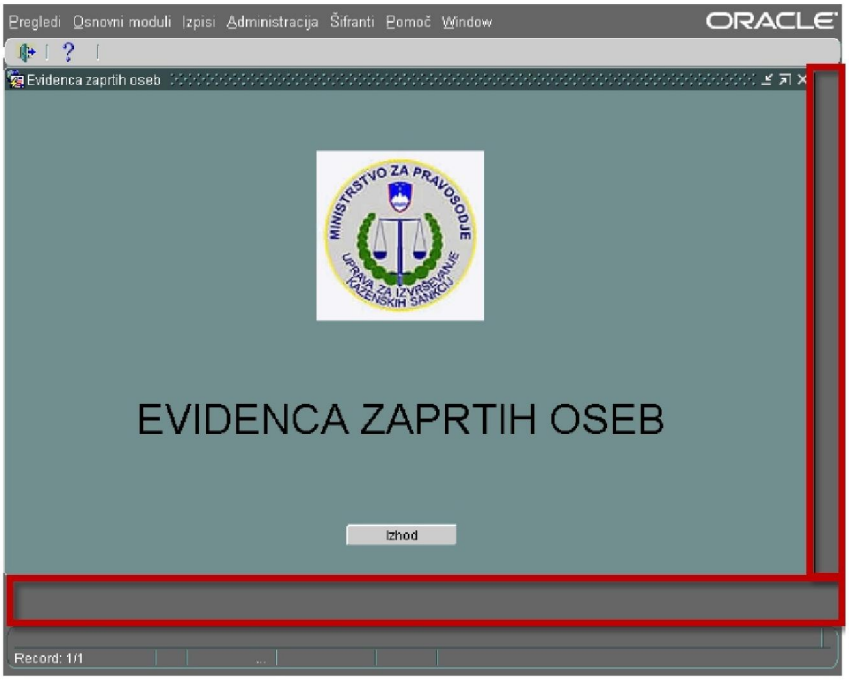
6. SPREMEMBA IMENA APLIKACIJE

Potrebno je spremeniti naziv aplikacije na vstopni maski v »Intranetna evidenca kaznovanih oseb«.



7. VELIKOST DELOVNE POVRŠINE

Potrebno je spremeniti višino in širino delovne površine, tako da bo enake velikosti kot osnovne maske v aplikaciji.



8.2.6 MANJŠE SPREMEMBE

SPECIFIKACIJA

Manjše spremembe na enotni maski za sprejem kaznovane osebe

V S E B I N A

ZGODOVINA DOPOLNITEV DOKUMENTA

Ime dokumenta	Datum zadnje spremembe	Avtor spremembe	Opis spremembe
SP-Manjše spremembe.doc	10.03.2010	Grega Rožac	Nov dokument.

1 ZAHTEVE

a. Zahtevana je sprememba oblike matične številke ob njenem avtomatičnem generiranju. V primeru mladoletnih obsojencev naj se uporabi oznaka MO, pri sodno pridržanih pa SP.
b. Zahtevana je dopolnitev na enotni maski, pri sklopu pozivi, na polju sodišča. Dovoljen je vnos samo okrožnih sodišč. Prav tako naj se v izbirnem oknu ponudijo samo okrožna sodišča.
c. Ob vnosu pripora na enotni maski naj se, v primeru da ima oseba čakajočo kazen ali če je na prekinitvi, prikaže obvestilo . Prav tako naj se ustrezno obvestilo prikaže na maski za prestajanja.
d. Iz enotne maske je zahtevan umik polj »čakajoča« in »umaknjena«.
e. Zahtevana je dopolnitev na enotni maski. Dovoljen je samo vnos sodbe, ki lahko nosi prestajanja. Prav tako naj se v izbirnem oknu ponudijo samo sodbe, ki nosijo prestajanja.

1.1 REFERENCE

Pošiljatelj - pobudnik	Projektna skupina e-pravosodje
Datum	20.04.2010 – 22.4.2010
Zadeva	Šolanje uporabnikov

2 ZASNOVA REŠITVE

2.1 PRAVILA

a. Na enotni maski za sprejem kaznovane osebe in na maski za prestajanja je potrebno dopolniti postopek za avtomatično generiranje matične številke:

- V primeru mladoletnih obsojencev naj bo matična številka v obliki MO-številka/leto
- Pri sodno pridržanih naj bo matična številka v obliki SP-številka/leto

b. Na enotni maski za sprejem kaznovane osebe, pri sklopu pozivi, je potrebno dopolniti vnos sodišča. Dovoljen naj bo samo vnos okrožnih sodišč. Prav tako naj se v izbirnem oknu ponudijo samo okrožna sodišča, ob napačnem vnosu pa naj se pokaže okno z ustreznim obvestilom.
Vrsta sodišča je zapisana v šifrantu sodišč.

c. Na enotni maski za sprejem kaznovane osebe in na maski za prestajanja je potrebno dodati kontrolo, ki ob vnosu pripora preveri ali ima oseba čakajočo kazen, ali je na prekinitvi. Ob shranjevanju naj se pojavi okno z ustreznim obvestilom.

d. Na enotni maski za sprejem kaznovane osebe je potrebno odstraniti polja »čakajoča« in »umaknjena«.

e. Na enotni maski za sprejem kaznovane osebe je potrebno dopolniti vnos sodbe. Dovoljen naj bo samo vnos sodb, ki lahko nosijo prestajanja. Prav tako naj se v izbirnem oknu ponudijo samo sodbe, ki lahko nosijo prestajanja, ob napačnemk vnosu pa naj se pokaže okno z ustreznim obvestilom.
Iz šifranta »vrste dokumentov« je razvidno ali določen dokument nosi prestajanje.

Rok za izdelavo:

5 delovnih dni po potrditvi specifikacije.

8.2.7 KONTROLA PRVEGA ZNAKA

SPECIFIKACIJA

Kontrola prvega znaka

V S E B I N A

ZGODOVINA DOPOLNITEV DOKUMENTA

Ime dokumenta	Datum zadnje spremembe	Avtor spremembe	Opis spremembe
SP-Kontrola prvega znaka.doc	10.03.2010	Aleš Kravos	Nov dokument.
SP-Kontrola prvega znaka v2.doc	15.03.2010	Aleš Kravos	Popravek – Opravilna številka se kontrolira na maski za vnos dokumenta in ne na maski za vnos kazni

1 ZAHTEVE

Zahtevana je kontrola na vnosu imena, priimka in opravilne številko sodbe, ki preverja, če je prvi vneseni znak presledek ali številka. V kolikor se to zgodi, potem mora biti uporabnik opozorjen.

1.1 REFERENCE

Pošiljatelj - pobudnik	Projektna skupina e-pravosodje
Datum	09.03.2010
Zadeva	Sestanek – pregled analize razlik in potrditev popravkov

2 ZASNOVA REŠITVE

2.1 PRAVILA

Na enotni maski za sprejem kaznovane osebe, maski za vnos osebe in maski za vnos dokumenta je potrebno pripraviti kontrolo na sledečih poljih:

- Priimek (Vnos osebe, enotna maska za sprejem)
- Ime (Vnos osebe, enotna maska za sprejem)
- Opravilna številka (Vnos dokumenta, enotna maska za sprejem)

Kontrola preverja, če je na prvem mestu besedila vnesen presledek ali številka. Torej sledeči znaki (' ','1','2','3','4','5','6','7','8','9','0').

V tem primeru mora aplikacija preklicati shranjevanje in uporabnika opozoriti na napako pri vnosu.

Rok za izdelavo:
01.04.2010 – Funkcionalnost nameščena na testnem okolju

9 DODATEK B - PRIMER TESTNEGA SCENARIJA ZA PROJEKT EZO

Št. scenarija: A2 - Sprejem gojenca - dolžina ukrepa 3 leto - odpust po prestani kazni	
Opis: Sprejem gojenca - dolžina ukrepa 3 leta - odpust po prestanem ukrepu	
Prijavitelj: RRC	**Datum:** 23.03.2010

Kratek povzetek postopka in problematike:
Referent vnese novo osebo, ki je sprejeta v zavod.

Proces podprt v iEZO aplikaciji:
Izpolni izvajalec RRC Računalniške storitve d.d.

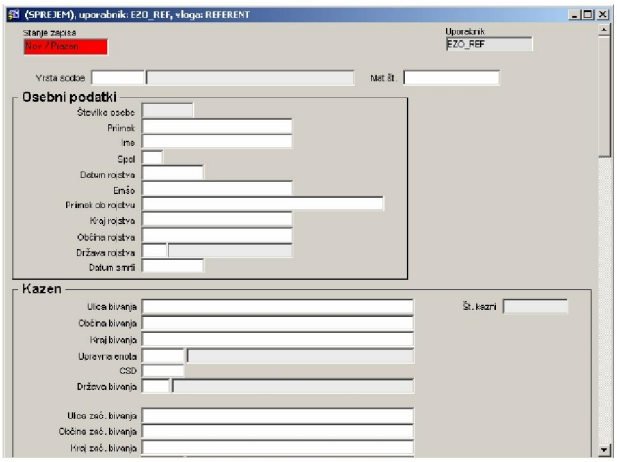
Seznam oseb v zavodu

Najprej v glavnem meniju odpremo seznam vseh zaprtih oseb:

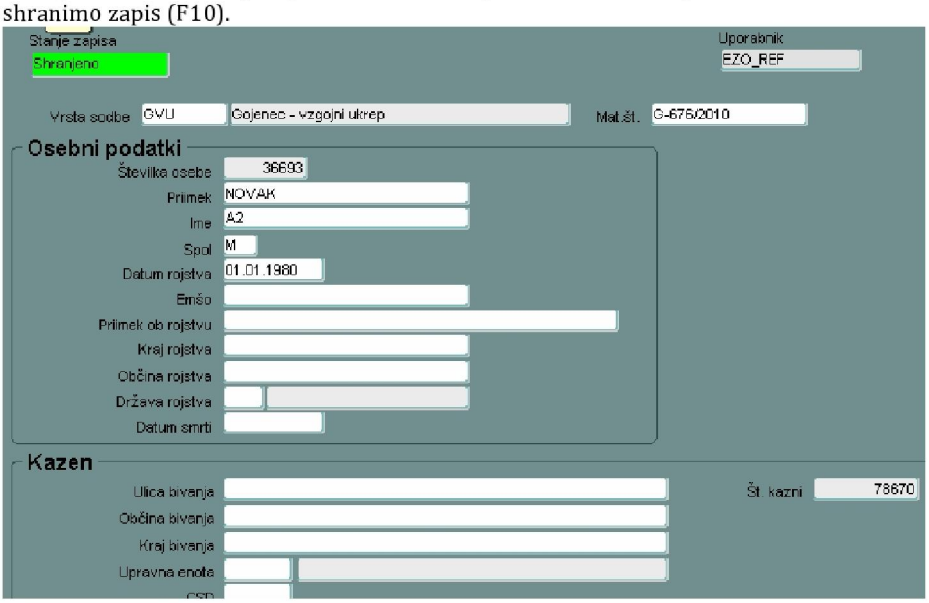
Odpre se zaslonska maska:

Sprejem nove osebe

Kliknemo na gumb »Sprejem nove osebe« levo spodaj. Odpre se naslednje okno:

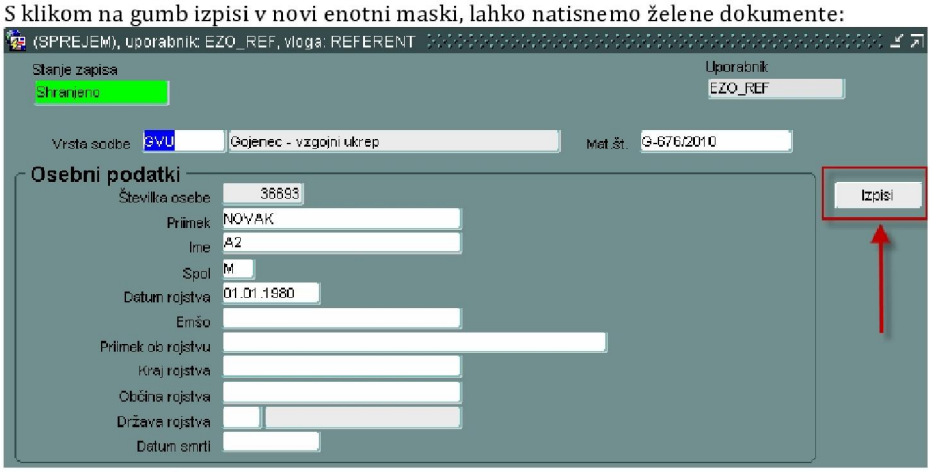


Pritisnemo tipko F9 in iz seznama izberemo vrsto sodbe. Matična številka se generira avtomatično ob shranitvi, lahko pa jo v izjemnih primerih vnesemo ročno. Ko se pomaknemo do polja »Priimek« lahko iz seznama izberemo obstoječo osebo, lahko pa vpišemo novo. Ko izpolnimo vse znane podatke, shranimo zapis (F10).

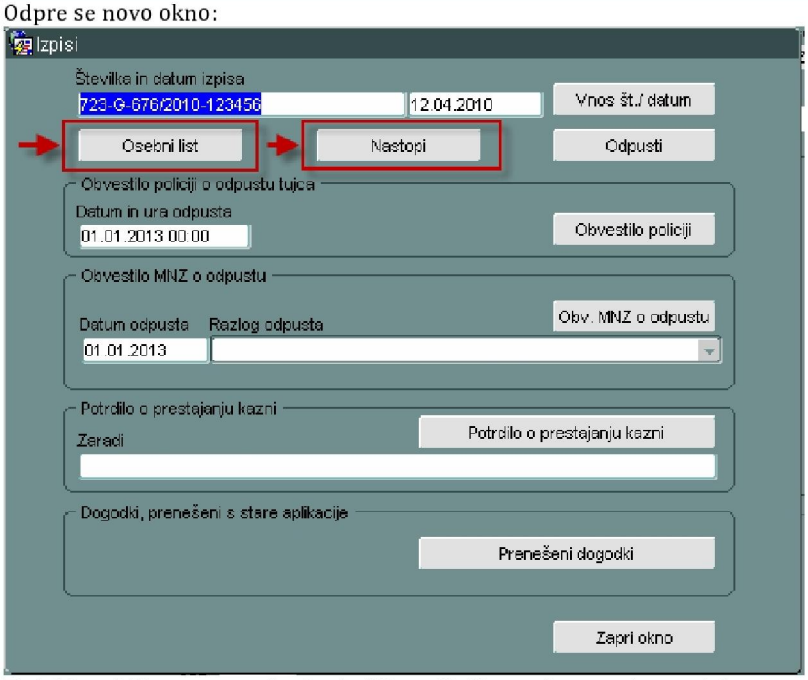


Izpis osebnega lista in nastopa

S klikom na gumb izpisi v novi enotni maski, lahko natisnemo želene dokumente:

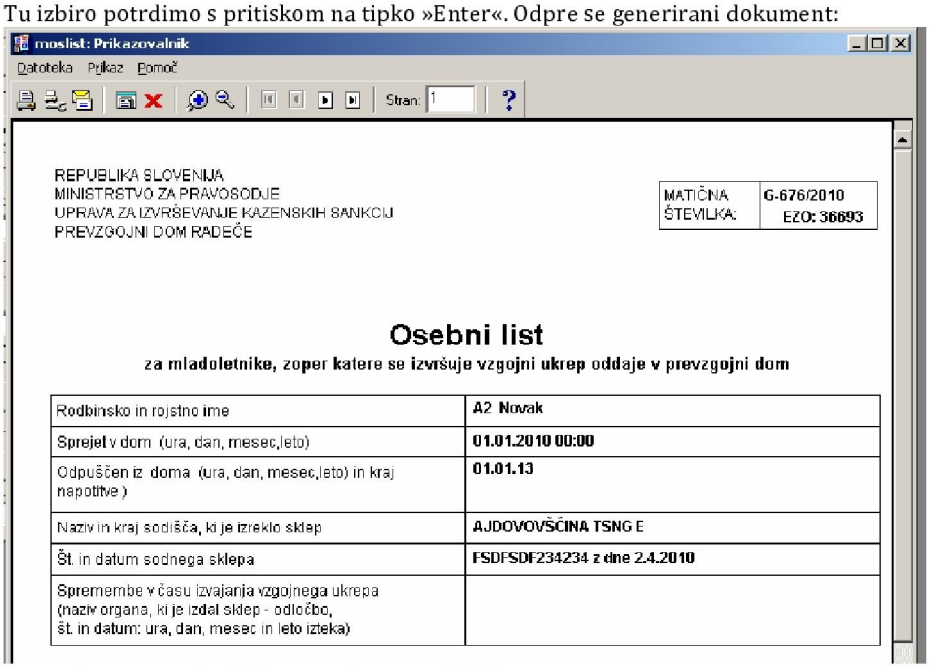


Odpre se novo okno:

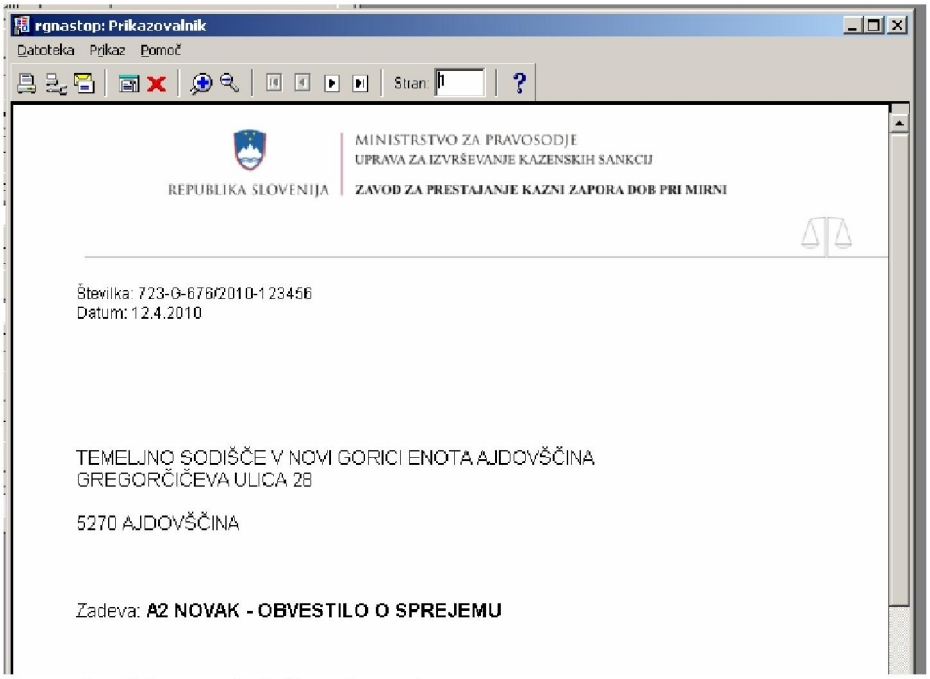


Tu lahko s klikom na gumb »Osebni list« ali »Nastopi« generiramo dokument, pripravljen za tiskanje.

Tu izbiro potrdimo s pritiskom na tipko »Enter«. Odpre se generirani dokument:



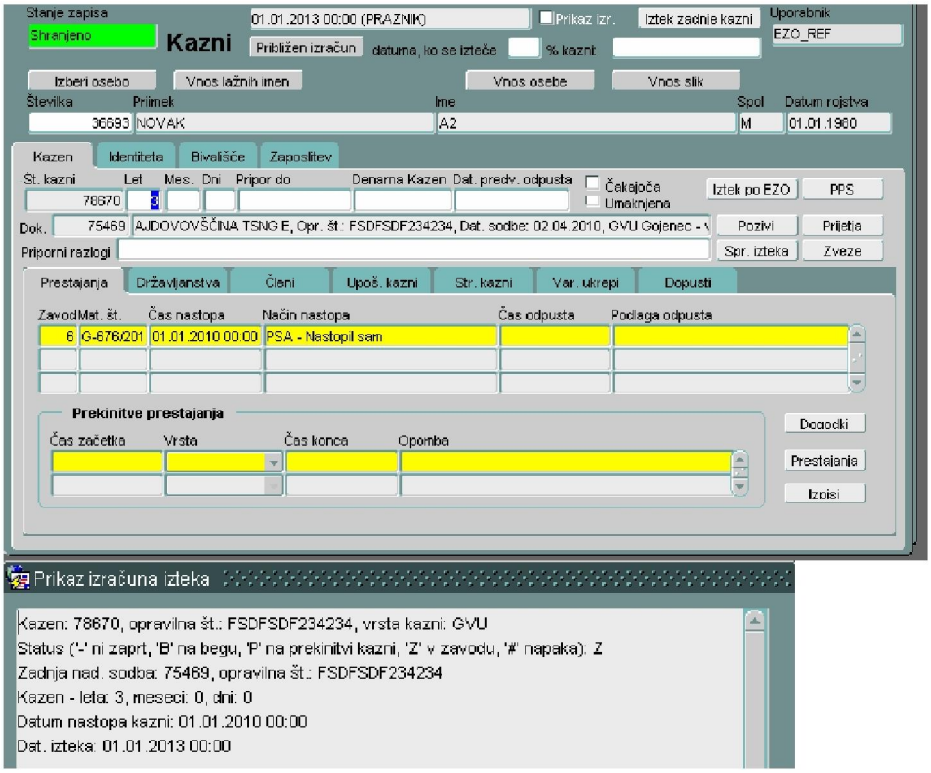
Po enakem postopku lahko generiramo tudi dokument za nastop:



Ko smo končali opravilo, lahko odprta okna zapremo.

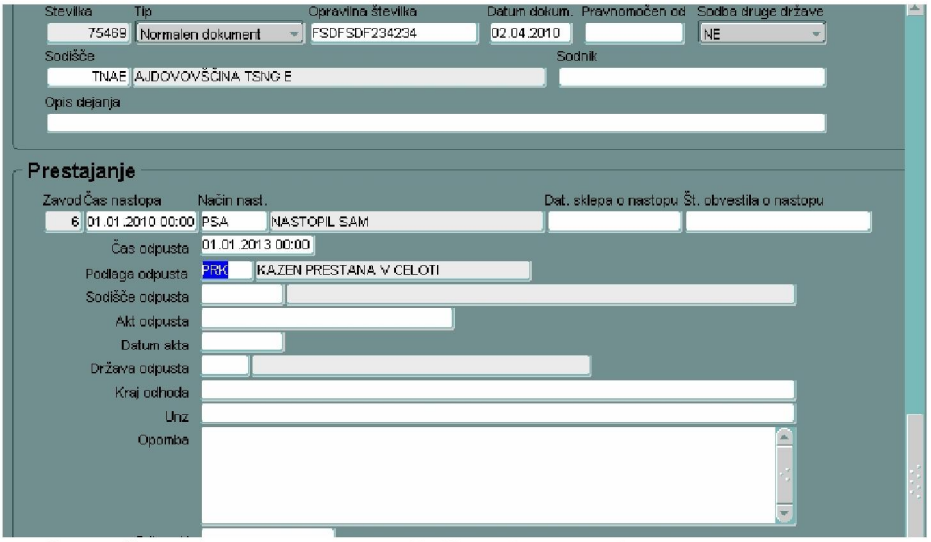
Izračun izteka kazni

Po želji lahko preverimo datum izteka kazni s klikom na gumb »Kazen« za izbrano osebo, kjer v oknu »Kazni« kliknemo na gumb »Iztek zadnje kazni« ter odkljukamo »Prikaz izr.«. Izpiše se naslednje okno:



Vnos datuma odpustitve

V maski »Seznam oseb v zavodu« do katere pridemo po postopku iz 1. koraka poiščemo osebo, ki jo želimo odpustiti iz zavoda (lahko uporabimo iskanje s tipko F7). Nato kliknemo na gumb »Ažuriranje« ter vnesemo »Čas odpusta«:



V polje »Podlaga odpusta« vnesemo PRK - KAZEN PRESTANA V CELOTI.
Vrednost lahko poiščemo na seznamu (F9).
Spremembe je potrebno shraniti (F10).

Obrazložitev postopka in problematike s strani naročnika – projektna skupina: Datum odgovora:

10 SLIKE

11 TABELE

12 LITERATURA

[1] John D. Gannon, James M. Purtilo, Marvin V. Zelkowitz, "Software Specification: A Comparison of Formal Methods", Department of Computer Science, University of Maryland, 2001

[2] Mike Cohn, "User Stories Applied", New York: Addison-Wesley, 2008

[3] Alistair Cockburn, "Use Cases, Ten Years Later", STQE Magazine, Apr. 2002, dostopno na naslovu: http://ac.cockburn.us/2098

[4] Kurt Bittner, Ian Spence, "Use Case Modeling", Addison-Wesley, 2003

[5] Robert Vienneau, "A Review of Formal Methods", Kaman Sciences Corporation, 1993

[6] Richard H. Thayer, Merlin Dorfman, "Software Requirements Engineering", IEEE Computer Society Press, California, 1997

[7] John M. Carrol, "Making Use: Scenario-based design in human-computer interaction", Cambridge, Mass.: The MIT Press, 2000

[8] IEEE Computer Society, "IEEE Recommended Practice for Software Requirements Specifications", New York, 1998

[9] Ron Jeffries, "Essential XP: Card, Conversation, and Comfirmation", XP Magazine, 30. Avgust, 2001

[10] C. Mazza, J. Fairclough, B. Melton, D. de Pablo, A. Scheffer, R. Stevens, "Software engineering standards", Prentice Hall International (UK) Limited, 1994

[11] Ivar Jacobson, "Object-Oriented Software Engineering", Addison-Wesley, 1992

[12] Kent Beck, Martin Fowler, "Planning Extreme Programming", Addison-Wesley, 2000

[13] Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum", Prentice Hall, 2002

[14] William C. Wake, "INVEST in Good Stories, and SMART Tasks", dostopno na naslovu: http://xp123.com/xplor/xp0308, 2003

[15] Tom Poppendieck, "The Agile Customer's Toolkit", Poppendieck LLC, 2003, dostopno na naslovu: http://www.torak.com/site/files/The%20Agile%20Customer%E2%80%99s%20Toolkit.pdf

[16] Alistair Cockburn, "Writing Effective Use Cases", Addison-Wesley, 2001