UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Jamnik

Razvoj prototipov v aplikaciji z orodjem Microsoft Expression Blend 3

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Matjaž Kukar

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: 00499/2010
Datum: 15.03.2010

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

Kandidat: **ANŽE JAMNIK**

Naslov: **RAZVOJ PROTOTIPOV APLIKACIJ Z ORODJEM MICROSOFT EXPRESSION BLEND 3**

APPLICATION PROTOTYPE DEVELOPMENT WITH MICROSOFT EXPRESSION BLEND 3

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Pomemben del projektnega dela pri razvoju programske opreme predstavlja izdelava aplikacije po želji in meri naročnika. V svojem diplomskem delu naj kandidat predstavi orodje za razvoj aplikacij aplikacij z orodjem Microsoft Expression Blend 3. Kandidat naj opiše postopke izdelave prototipa spletnih in namiznih aplikacij in osvetli morebitne razlike. Opiše naj možnosti komuniciranja pripomb in popravkov med naročnikom in razvijalcev. Opisane značilnosti orodja naj ilustrira s primeri praktične uporabe pri razvoju vzorčne spletne in namizne aplikacije.

Mentor:
doc. dr. Matjaž Kukar

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Franc Solina

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

Spodaj podpisani **Anže Jamnik**,

z vpisno številko **63040217**,

sem avtor diplomskega dela z naslovom:

Razvoj prototipov aplikacij z orodjem Microsoft Expression Blend 3

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc.dr. Matjaža Kukarja
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne: 15.09.2010 Podpis avtorja:

Zahvala

Zahvalil bi se doc. dr. Matjažu Kukarju za strokovno pomoč in vodenje pri izdelavi diplomske naloge. Zahvalil bi se tudi mojemu šefu v podjetju B2 d.o.o., ker mi je pomagal izbrati temo za diplomsko nalogo in priskrbel literaturo. Zahvalil bi se tudi vsem, ki so kakorkoli pomagali pri diplomski nalogi. Posebna zahvala pa gre moji družini, ki mi je omogočila študij in me vseskozi podpirala.

Kazalo

Povzetek 1
Abstract 2
1. Uvod 3
2. Osnove projektnega dela 4
2.1. Splošno 4
2.2. Izhodišča za delo 4
2.3. Financiranje dejavnosti razvoja 5
2.4. Vloge 5
3. Štiri faze projekta 6
3.1. Vizija 6
3.1.1. Splošno 6
3.1.2. Kdo jo izdela? 7
3.1.3. Metode dela 7
3.1.4. Komunikacija in Expression Blend 3 8
3.1.5. Mejnik odobritev vizije ter izdelki faze vizije 8
3.2. Planiranje 11
3.2.1. Splošno 11
3.2.2. Kdo ga izvede? 12
3.2.3. Metode dela 12
3.2.4. Konceptualno načrtovanje 13
3.2.5. Logično načrtovanje 15
3.2.6. Fizično načrtovanje 15
3.2.7. Mejnik odobritev projektnega plana in izdelki faze planiranja 20
3.3. Razvoj 23
3.3.1. Splošno 23
3.3.2. Kdo ga izvede? 24
3.3.3. Metode dela 24
3.3.4. Implementacija 25
3.3.5. Mejnik zaključen razvoj ter izdelki faze razvoja 25
3.4. Stabilizacija 28
3.4.1. Splošno 28
3.4.2. Kdo jo izvede? 29
3.4.3. Metode dela 29
3.4.4. Mejnik izdaja različice ter izdelki faze stabilizacije 30
3.5. Dostava produkta 31
3.6. Programska orodja, ki pomagajo pri razvoju aplikacij 31
3.6.1. Microsoft Visio 32
3.6.2. Microsoft Expression Blend 32
3.6.3. Microsoft Project 32
3.6.4. Microsoft Visual Studio 32
4. Expression Blend 3 – enostavna rešitev za izdelavo prototipov 33
4.1. Expression Blend 4 in njegove novosti 34
4.2. Prvi stik z novim okoljem in ureditev okolja na delo 34
4.2.1. Predstavitev zavihkov 36
4.3. Predstavitev SketchFlow player-ja 39
4.4. Izdelava prototipa preproste spletne aplikacije 40
4.4.1. Kratek opis aplikacije 40
4.4.2. SketchFlow map in nodi 40

4.4.3. Komponentna okna 41
4.4.4. Kako do komponentnega okna 42
4.4.5. Gumbi in akcije 42
4.4.6. Iskalnik in xml podatki 42
4.4.7. Okno Opis in primer opisa filma 43
4.4.8. Animacija iskanja in izbira filma 44
4.4.9. Predloga 47
4.4.10. Izvoz projekta in povratne informacije 48
4.4.11. Povratne informacije (Feedback) 49
4.5. Preprosta spletna aplikacija 53
4.5.1. Slike končne aplikacije 53
4.6. Preprosta WPF prototipna namizna aplikacija 56
4.7. Pretvorba WPF prototipa v WPF aplikacijo. 56
4.7.1. Prikaz pretvorbe WPF prototipa v WPF aplikacijo 56
5. Zaključek 60
Kazalo slik 61
Kazalo tabel 63
Viri in literatura 64

Seznam kratic in simbolov

API	Application programming interface
MS PowerPoint	Microsoft PowerPoint
MSF	Microsoft Solutions Framework
RHB	Različica brez hroščev
SQL	Structured Query Language
SUBP	Sistem za upravljanje baze podatkov
TFS	Team Foundation Server
UML	Unified Modeling Language
UPW	Unified Process Workflows
VS2010	Microsoft Visual Studio 2010
WPF	Windows Presentation Framework
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
ZBR	Zero Bug Release

Povzetek

V diplomski nalogi sem se osredotočil na prikaz poteka izdelave aplikacije. Predstavil sem vse faze projekta, ki so potrebne za dosego le tega. Predstavil sem tudi načine, kako je mogoče olajšati vloženo delo ter skrajšati čas, ki je potreben za zaključek nekega projekta. V ta namen sem uporabil orodje Microsoft Expression Blend 3. S pomočjo tega orodja lahko izdelujemo prototipe aplikacij in jih istočasno uporabimo tudi pri dejanski aplikaciji. V aplikaciji je dobro poskrbljeno tudi za komunikacijo. Stranka in razvijalec si lahko izmenjujeta informacije na enostaven način.

V diplomski nalogi sem predstavil orodje Expression Blend 3 ter izdelavo enostavnega prototipa s tem orodjem. Prototip sem naredil s pomočjo tehnologije Silverlight, ki je namenjena spletnim aplikacijam. Celoten prototip je narejen brez vnašanja kode. V okviru prototipa sem predstavil potek komunikacije. Sprogramiral sem še končno aplikacijo v okolju Microsoft Visual Studio 2010 s pomočjo C# in .NET. Tudi izgled končne aplikacije je prikazan v diplomi. Aplikacija ima v ozadju bazo podatkov (SQL Server 2008). Predstavil sem še preprosto namizno aplikacijo, ki temelji na WPF tehnologiji. Pri namizni aplikaciji sem prikazal, kako se prototip aplikacije pretvori v dejansko aplikacijo. Po pretvorbi se je ohranilo vse.

Ključne besede: Microsoft Expression Blend 3, Microsoft Visual Studio 2010, SQL Server 2008, C#, .NET, prototip, aplikacija, komunikacija, Silverlight, WPF, XAML.

Abstract

The BA thesis focuses on the process of application building. All project phases required to build an application as well as methods for facilitating the work and shortening the time needed to complete a certain project are presented. With the help of Microsoft Expression Blend 3 the application prototypes can be developed and used with an actual application at the same time. The exchange of information between the customer and the developer is simple as good communication is a key factor in the application.

The BA thesis also presents the Expression Blend 3 and the development of a simple prototype using this tool. The prototype was developed using the Silverlight technology which is designed for web applications. The entire prototype was developed without writing any code. The process of communication is presented within the framework of the prototype. The final application was programmed in Microsoft Visual Studio 2010 environment using C# and .NET. The appearance of the final application is also shown. The application is using a SQL Server 2008 database. The BA thesis presents a simple desktop application based on the WPF technology and the process of transformation from a prototype application to an actual application. After the transformation there was no change.

Keywords: Microsoft Expression Blend 3, Microsoft Visual Studio 2010, SQL Server 2008, C#, .NET, prototype, application, communication, Silverlight, WPF, XAML.

1. Uvod

Pisanje projektov ni muha enodnevnica in je za marsikaterega posameznika ali za podjetje velikega pomena. Vključuje veliko znanj in veščin, ki si sproti pridobivamo. Vsa podjetja razpolagajo z omejenim številom virov, kot so ljudje, material in stroški. Projektni vodje imajo omejeno število virov, ki jih lahko uporabijo na projektih. Zato je pomembno, da je vsaka faza projekta čim hitreje in čim bolj kakovostno zaključena. Problematičen del vsakega projekta je komunikacija med strankami in razvijalci. Težko se je sporazumeti, kakšne so želje stranke. Komunikacija in izdelava prototipov predstavlja velik problem, tako da sem se v diplomski nalogi lotil raziskovanja tega področja. [3]
Glavni cilj diplome je prikazati potek projekta od ideje do končne delujoče aplikacije in pa predstaviti orodje Expression Blend 3. S tem orodjem lahko enostavno gradimo prototipe aplikacij in rešujemo problem komunikacije. Prikazal bom potek izdelava prototipa spletne aplikacije. V okviru spletne aplikacije bom predstavil tudi način komunikacije v tem orodju. Predstavil bom tudi primer namiznega prototipa, pri katerem bom prikazal postopek pretvorbe prototipa v aplikacijo.
V drugem in tretjem poglavju je bralcu predstavljen projekt na splošno ter faze projekta, ki so potrebne za uspešno vodenje projekta. Znotraj posameznih faz projekta je natančno predstavljen namen faze, kdo jo izdela, katere metode dela so potrebne, na kak način poteka komunikacija in kakšni so mejniki za odobritev faze.
Četrto poglavje in glavni del diplomskega dela je namenjen predstavitvi orodja Expression Blend 3 in izdelavi prototipov. Expression Blend ima na voljo dve platformi. Ti dve platformi sta WPF in Silverlight. Za kodiranje pa imamo na voljo dva .NET jezika, C# in Visual Basic, ter opisni programski jezik XAML za dizajn.
WPF in Silverlight sta platformi, ki sta uporabni v zahtevnih aplikacijah z uporabo 3D, interaktivnosti, multimedije in prilagojenimi uporabniškimi vmesniki. Trenutno najboljša kombinacija za hiter razvoj WPF in Silverlight aplikacij je Microsoft Visual Studio 2010 in Microsoft Blend 4.
Iz tega je vidna težnja Microsofta po ločenosti oblike programa, ki jo naredi oblikovalec v Expression Blend 4, od kode oziroma poslovne logike programa, ki jo spiše programer v Visual Studiu. Prednost teh dveh platform je tudi sam opisni jezik XAML, ki je narejen po standardih XML jezika. S tem lažje ločimo oblikovni del aplikacije od programerskega dela, seveda pa kasneje lažje uporabimo že napisano XAML kodo (ti. predlogo). Ker so fizično ločene tudi datoteke, lahko oblikovalec in programer delata istočasno na istih datotekah. Tu je dobro omeniti povezljivost z Silverlight aplikacijami, saj si WPF in Silverlight delita iste osnove, npr. XAML. WPF in Windows Forme sta tudi popolnoma kompatibilna, saj lahko v Windows Forme vstavljamo elemente WPF in obratno.
XAML je osnova za WPF in Silverlight grafiko. Uporablja se za vektorske slike, stile, barvne palete, grafične vmesnike, ustvarjanje dokumentov za tisk in še mnogo drugih stvari. [4]

2. Osnove projektnega dela

2.1. Splošno

Projekt je enkratna, praviloma zahtevna in kompleksna skupina nalog, ki mora biti končana v določenem roku, doseči mora vnaprej določene in morebitne kasneje odkrite cilje, ter upoštevati vse podane in kasneje odkrite omejitve.
Projekt je ciljno usmerjen in zaključen proces razvijanja dejavnosti, ki so usmerjene k doseganju končnega cilja. Do tega cilja se prihaja postopoma z doseganjem posameznih podciljev. Pobuda za projekt lahko vsebuje samo en končni cilj ali pa vse podcilje. Pobudnik projekta je lahko posameznik, podjetje, družbena organizacija, javna organizacija, družbena institucija, državna institucija ali celo mednarodna organizacija, ki je običajno tudi naročnik projekta. Naročnik projekta običajno postavi tudi cilje projekta. Za uspešno realizacijo projekta je potrebno poiskati še izvajalce in določiti vodje projekta. Vodstvo ima pri organiziranju, upravljanju in vodenju projekta odločilno vlogo. Vodstvo projekta mora načrtovati in aktivirati posamezne aktivnosti ter jih tehnično, časovno in finančno uskladiti. Projekti so lahko majhni ali veliki, vsi pa morajo biti časovno omejeni, predviden mora biti čas začetka in zaključka. Uspešno izvajanja projekta zahteva tudi dobro postavljen informacijski sistem. [5]
Za planiranje in vodenje projektov si pomagamo z razvojno procesnim modelom MSF (Microsoft Solution Framework).

2.2. Izhodišča za delo

Osnovno izhodišče za aktivnosti na področju razvoja je projekt. Vsaka aktivnost, ki se izvaja na področju razvoja, tako pripada določenemu projektu. Projekt se lahko oblikuje na podlagi:

- *Prodaje (zahteva po pripravi ponudbe, zahteva po pripravi analize, …),*
- *stranke in*
- *vodje razvoja na podlagi plana razvoja.*

V sklopu vsakega projekta se na področju razvoja izvajajo naslednje faze:

- *Vizija,*
- *planiranje,*
- *razvoj in*
- *stabilizacija.*

Posamezno nalogo v določeni fazi lahko odredijo naslednje vloge v projektu:

- *Produktno upravljanje: vizija, načrtovanje,*
- *razvoj: razvoj,*
- *projektno upravljanje: načrtovanje, razvoj, stabilizacija,*
- *testiranje: stabilizacija,*
- *vzdrževanje: stabilizacija.*

Posamezne naloge se izvajajo po pravilih, opisanih v poglavjih v nadaljevanju in v skladu s Shemo razvoja. [2]

2.3. Financiranje dejavnosti razvoja

Dejavnosti razvoja se lahko financirajo s prodajo rešitev ter iz rezerv oddelka IS.
Viri financiranja so:

- kupnine od prodanih programov,
- plačljiva dela, opravljena za stranke, in
- vzdrževalne pogodbe. [2]

2.4. Vloge

Za vsak projekt se vzpostavi projektna skupina, ki je razdeljena v naslednje vloge:

- *Produktno upravljanje* (kupčev zagovornik v timu in obratno, skrb za produktne dokumente, zmogljivosti produkta, prioritete razvoja ter komunikacijo),
- *projektno upravljanje* (odgovornost za dostavo pravega produkta ob pravem času, skrb za pravi obseg produkta tudi v finančnem smislu),
- *razvoj* (izdelava in testiranje, ocena časa in napora za realizacijo, svetovanje),
- *testiranje* (določitev strategije testiranja, testiranje, preverjanje kvalitete) in
- *vzdrževanje* (določitev strategije namestitve, izvedba namestitve, upravljanje z nadgradnjami).

Na manjših projektih (manj od 500 ur) sta vlogi *produktno* in *projektno upravljanje* združeni (kot vloga *vodja projekta*). Prav tako ni nujno, da obstaja vloga *vzdrževanja*, ampak so v tej vlogi lahko člani iz vlog *razvoj* in *testiranje*. Ne glede na velikost projekta pa ni dopustno, da zgolj člani vloge *razvoj* izvajajo naloge testiranja. [2]

3. Štiri faze projekta

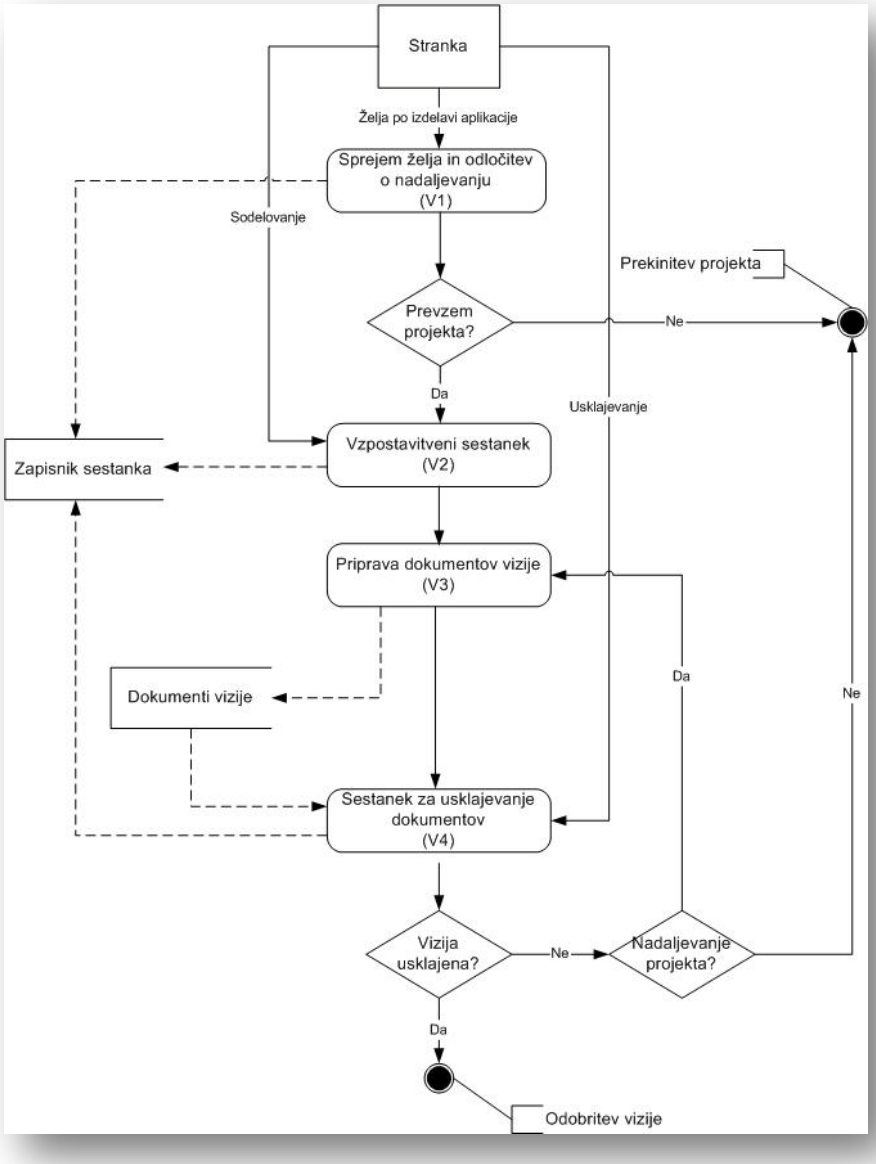
Projektne faze delimo na štiri dele:

- *Vizija,*
- *planiranje,*
- *razvoj in*
- *stabilizacija.*

3.1. Vizija

3.1.1. Splošno

Pred začetkom kateregakoli dela projekta mora tim izdelati *vizijo*, ki opisuje skupni cilj in smer projekta. Med samo izdelavo vizije mora imeti tim svobodo za »brainstorm« vseh elementov projekta. Priprava in usklajevanje dokumenta vizije omogoča timu in stranki zbistritev ciljev projekta in omogoči, da vsi vidijo, kaj je namen zaključenega projekta.
Vizija predstavlja približno 10 do 15% celotnega projekta. [2]



Slika 1: Shema vizije

3.1.2. Kdo jo izdela?

Spodnja tabela opisuje tipične zadolžitve vlog v fazi *vizije*. Vodja vsake vloge skrbi za izvršitev zadanih nalog in komunikacijo z ostalimi člani tima.

Tabela 1: Tipične zadolžitve vlog v fazi vizije

Vloga	Zadolžitve
Produktno upravljanje	Dostavi dokument vizije. Upravlja zahteve stranke. Vključuje stranko pri prototipnem razvoju. Upravlja z tveganji projekta
Projektno upravljanje	Izdela cilje dizajna. Opiše koncept rešitve. Določi okvirje projektne strukture. Upravlja s tveganji projekta
Razvoj	Načrtuje prototipe. Opiše okvire možnosti razvoja. Identificira posledice.
Uvajanje	Identificira uporabniške zahteve po učinkovitosti in posledice. Upravlja s pričakovanji uporabnikov. Vključuje uporabnike pri prototipnem razvoju. Upravlja s tveganji projekta.
Testiranje	Specificira kriterije odobritve. Skicira sistem za odkrivanje hroščev. Skicira sistem za upravljanje s tveganji. Identificira tveganja projekta.
Logistično upravljanje	Identificira posledice razvoja. Identificira posledice podpore. Upravlja s tveganji projekta.

[6]

3.1.3. Metode dela

Pri pripravi vizije projekta se uporabljajo koraki, podobni UPW (Unified Process Workflows). Sledijo si v fleksibilnem zaporedju, ki omogoča vračanje v prejšnje korake. Koraki so: *raziskava, analiza, implementacija* in *validacija*. [2]

3.1.3.1. Raziskava

Za izdelavo vizije mora tim opraviti določene raziskave, kjer se osredotoči na stranko in na druge podobne produkte, razvite ali uporabljene s strani organizacije.

Cilj in izhod: Raziskati in zabeležiti je potrebno čim več informacij. Izvori teh informacij naj bodo stranka, uporabniki, trenutne aplikacije in njihova dokumentacija kakor tudi strokovnjaki iz obravnavanega področja. [2]

Raziskava drugih aplikacij podjetja

Če že obstaja kakšna predhodna različica produkta, naj se tim najprej seznani z vizijo predhodne različice. Velika verjetnost je, da predhodna različica vsebuje dolgoročne cilje, ki naj bi jih naslednje različice implementirale. Podobno pa mora tim razumeti tudi vizije drugih aplikacij, ki so trenutno v razvoju. Nekatera vprašanja, ki naj si jih tim zastavi ob pripravi vizije za nov produkt, so:

- Ali predstavlja nov produkt velik ali majhen korak za organizacijo?
- Ali sta zamišljen koncept in arhitektura nova za organizacijo?
- Katere novosti prinaša produkt za organizacijo? [2]

Raziskava stranke in uporabnikov

Faza Vizije predstavlja čas, ko se tim osredotoči na problematiko stranke in kako bodo uporabniki produkt uporabljali da bodo le to reševali. Področja, ki naj jih tim pregleda, so: poslovne pobude, obstoječe informacije, nova strojna oprema, nove tehnologije, problematika vzdrževanja, plani uvajanja uporabnikov, globalna problematika. [2]

Raziskava podobnih produktov
Vizija, ki jo tim izdela za produkt, mora biti usklajena z vizijami za že razvite produkte organizacije. Vprašanja, ki naj si jih tim ob tem zastavi, so naslednja: vpliv vizij ostalih produktov, vpliv operacijskega sistema na obstoječe produkte, urnik razvoja ostalih produktov za organizacijo v trenutnem obdobju. [2]

3.1.3.2. Analiza
Cilj: V *fazi analize* tim analizira obstoječe stanje s pomočjo informacij o poslovanju in informacij od uporabnikov. Ugotoviti je potrebno trenutne značilnosti.
Izhod: Izhodišče predstavljajo diagrami primerov uporabe in diagrami aktivnosti, pripravljeni po UML, ki najlažje predstavijo poslovne in uporabniške zahteve. V dodatno pomoč so tudi opisi specifičnih primerov uporabe, scenarijev primerov uporabe in diagramov aktivnosti. Ti diagrami predstavljajo osnovo za celotni proces razvoja. [2]

3.1.3.3. Racionalizacija
Po izdelanih primerih uporabe in seznamu značilnosti je potrebno podobne značilnosti združiti v večje sklope na podlagi uporabniških akcij, akcij aplikacij, uporabljenih podatkov ali drugih smiselnih grupiranj. V tej fazi se tim še vedno ukvarja z značilnostmi, ki opisujejo trenutno stanje, kakor tudi že značilnostmi bodočega stanja.
Cilj: Racionalizirati je potrebno izdelane modele v želji po doseganju večje učinkovitosti aktivnosti.
Izhod: Izhodišče predstavljajo racionalizirani primeri uporabe, njihovi scenariji in diagrami aktivnosti. [2]

3.1.3.4. Implementacija
V *fazi implementacije* tim pregleda predhodno definirane sklope značilnosti in jim določi prioritete. Skupaj so zbrani tudi primeri uporabe, njihovi scenariji in diagrami uporabe bodočega stanja, izdelan pa je tudi osnutek dokumenta vizije. Ta vsebuje dolgoročne cilje projekta. V tem trenutku je znanih dovolj podatkov, da je lahko izdelan, optimiziran in sprejet trikotnik variabilnosti virov, značilnosti in datuma zaključka. Prav tako je lahko izdelan okvirni projektni plan, ki se uporabi kot osnova za *fazo planiranja*.
Cilj: Izdelati je potrebno osnutek dokumenta vizije in projektnega plana.
Izhod: Izhodišče je osnutek dokumenta vizije in projektnega plana. [2]

3.1.3.5. Validacija
Cilj: Cilj validacije je revizija in dodelava dokumenta vizije. Prav tako se je potrebno ozreti na predhodne korake in izdelke ter ugotoviti pripravljenost tima na mejnik *vizija odobrena*.
Izhod: Rezultat validacije je revidiran in usklajen končni dokument vizije. [2]

3.1.4. Komunikacija in Expression Blend 3
Faza vizije ni samo zbiranje in priprava dokumentov, temveč odprta debata o viziji projekta projektnega tima in predstavnikov stranke. Prototipi predstavljajo učinkovit način komunikacije. V Expression Blend-u 3 imamo hiter in učinkovit sistem za komunikacijo. Z njim se lahko izognemo številnim sestankov. Več o tem je razloženo v poglavju, ki opisuje Expression Blend. [2]

3.1.5. Mejnik odobritev vizije ter izdelki faze vizije
Cilj *faze vizije* je doseg mejnika *odobritev vizije*, ki predstavlja dogovor oziroma pogodbo med stranko / naročnikom in timom o glavnih kritičnih izhodiščih projekta.
Za doseg mejnika *odobritev vizije* so potrebni naslednji izdelki:

- dokument vizije,
- dokument projektne skupine in
- glavni dokument tveganj.

Na tem mejniku se tim tudi odloči o nadaljevanju projekta ali njegovi opustitvi. Kajti dobro zaključena *faza vizije* lahko timu in stranki oziroma naročniku omogoča nadaljevanje projekta brez strahu pred neuspehom. [2]

3.1.5.1. Dokument vizije

Dokument vizije, ki je izdelan ob koncu *faze vizije*, je učinkovit, če vsebuje vsaj naslednje elemente:

- izjavo vizije,
- raziskavo uporabnikov,
- informacije o konkurenci,
- bodoče sklope značilnosti in
- grob projektni urnik. [1]

Definiranje izjave vizije

Izjava vizije predstavlja veliko lastnosti pametnega cilja, ki so:

- specifičnost (izrecen, svojstven),
- merljivost (lahko v vsakem trenutku izmerimo, kolikšen je napredek),
- izvedljivost,
- relevantnost in
- časovna definiranost.

Pravilo za jedrnato izjavo vizije je enostavno definirano in vsem razumljivo. [1]

Raziskava uporabnikov

Vse, kar tim z raziskavo pridobi od stranke in uporabnikov, predstavlja osnovo za nadaljnje korake. To upošteva raziskavo konteksta, raziskavo trga, ciljne skupine, … Raziskava tudi ponudi informacije za pripravo primerov uporabe, ki opisujejo trenutno in bodoče delo uporabnikov. [1]

Informacije o konkurenci

Tu si zastavimo vprašanje, katere druge aplikacije, ki ponujajo rešitev zahtev stranke, so še na trgu. Tim mora zagotoviti, da produkt, opisan z vizijo, rešuje zahteve stranke bolje kakor katerakoli druga obstoječa aplikacija in pri tem še upravičuje investicijo. [1]

Opis bodočih sklopov značilnosti

Sklopi značilnosti so kategorije, v katero bodo spadale vse bodoče značilnosti produkta. Programsko upravljanje uporabi te kategorije pri ugotavljanju, ali bodoče značilnosti zadovoljujejo cilje projekta. Če značilnost ne spada v noben sklop, najbrž ni pomembna za trenutno različico. [1]

Določitev prioritet in projektnega urnika

Po zaključku *faz raziskave*, *analize* in *racionalizacije* v *procesu vizije* ima tim grob načrt projektnega urnika in virov za njegovo uresničitev. Tim izdela ustrezen trikotnik variabilnosti, ki prikazuje porazdelitev virov, značilnosti in datum zaključka. [1]

3.1.5.2. Dokument projektne skupine

Dokument za razliko od dokumenta vizije, ki opisuje, kaj naj bo izdelano, opisuje, kdo naj to izdela. Dokument opisuje:

- vse vloge tima,
- kdo je vodja posamezne vloge,
- kdo so člani tima in njihove kontaktne informacije,
- kdo je naročnik projekta,
- kako bo projekt upravljan, npr. kontrola nad spremembami in način poročanja,
- urnik sestankov projekta, e-pošta in spletne strani. [1]

3.1.5.3. Glavni dokument tveganj

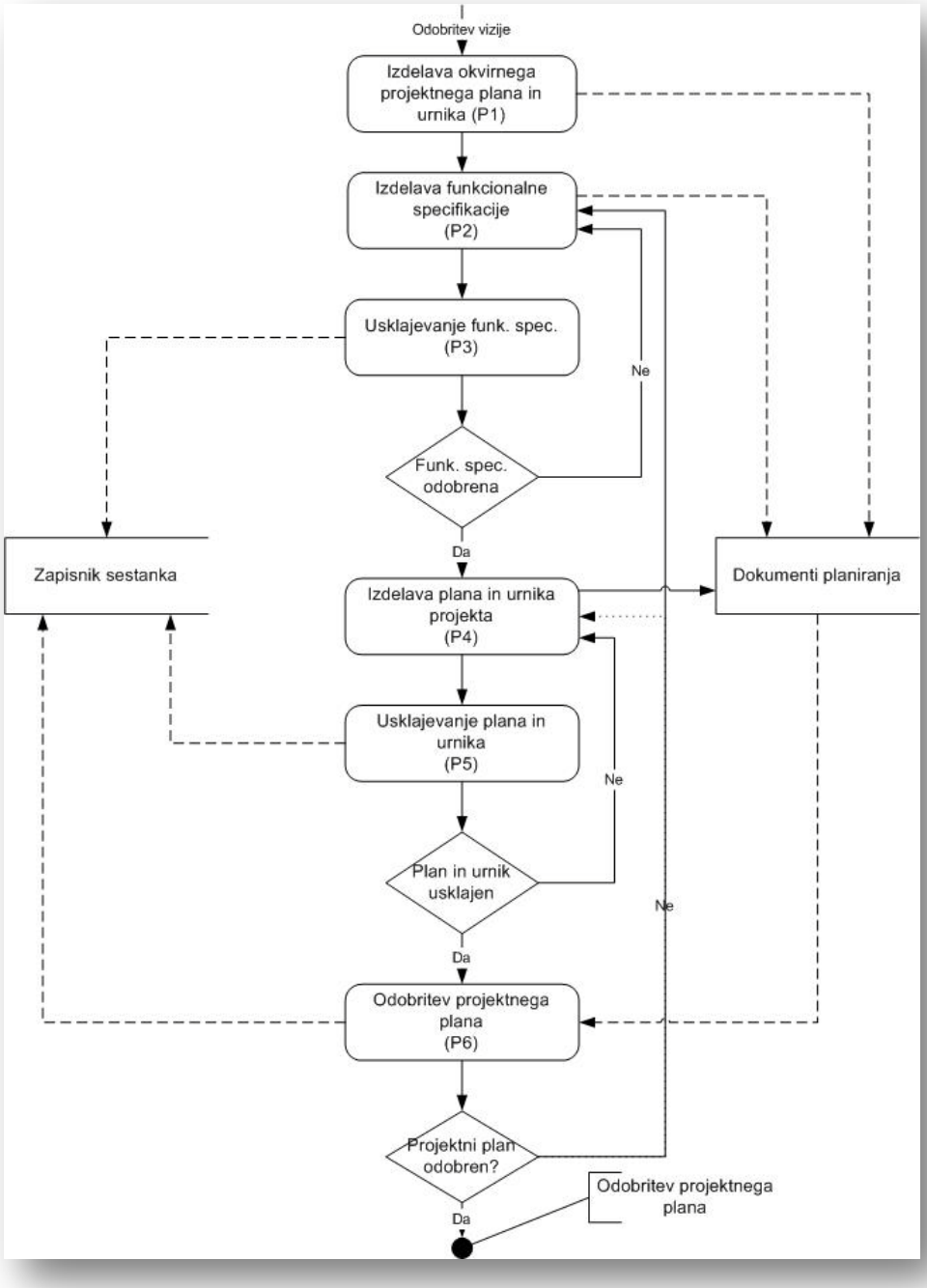
Dokument vsebuje analizo tveganj projekta ter plana za njihovo odpravo. Večji del dokumenta predstavlja spisek tveganj z opisi. Ta so največkrat predstavljena kot najpomembnejših deset tveganj. Plani odprave pa vsebujejo način odprave ter kdo je zadolžen za njihovo izvršitev. Prav tako je priporočljiv graf, ki predstavlja tveganja, njihovo verjetnost pojavitve in vpliva na projekt. [1]

3.2. Planiranje

3.2.1. Splošno

Medtem, ko je bilo v *fazi vizije* glavno vprašanje »Ali lahko s pomočjo tehnologije rešimo določen poslovni problem in če, kako?« in rezultat razumevanje problema in skupna vizija rešitve, je v *fazi planiranja* glavno vprašanje »Kaj v resnici potrebujemo, da dosežemo zastavljeno vizijo?«. Rezultat pa je podroben projektni plan, ki ga odobravata tako stranka kot tudi projektni tim. Tu se tudi zastavijo najpomembnejša vprašanja, ki so: »Katere značilnosti bodo počakale na naslednjo različico?«, »Katera tehnologija še ni zrela za uporabo?«, »Kateri so tisti stroški, ki ne upravičujejo razvoja?«, …

Planiranje predstavlja približno med 35 do 40% celotnega projekta. [2]



Slika 2: Shema planiranja

3.2.2. Kdo ga izvede?

Spodnja tabela opisuje tipične zadolžitve vlog v *fazi planiranja*. Vodja vsake vloge skrbi za izvršitev zadanih nalog in komunikacijo z ostalim člani tima.

Tabela 2: Tipične zadolžitve vlog v fazi planiranja

Vloga	Zadolžitve
Produktno upravljanje	Vodi proces zbiranja zahtev in proces konceptualnega načrtovanja. Dela na planu in urniku komunikacij.
Projektno upravljanje	Vodi celoten proces načrtovanja, predvsem logično načrtovanje. Izdela osnutek funkcionalne specifikacije. Prevzame vodstvo nad celoto in preverja, ali tim dosega plan in urnik.
Razvoj	Vodi fizični del načrtovanja funkcionalne specifikacije. Določi čas in obremenitev za izgradnjo in stabilizacijo produkta, ki ga specificira v planu in urniku razvoja. Razvije potrebne sisteme preverjanja ustreznosti koncepta planiranja (proof of concept).
Uvajanje	Analizira uporabniške potrebe. Izdela strategijo za podporo zmogljivostim. Oceni celotno načrtovanje s stališča uporabnosti. Določi čas in obremenitev za izgradnjo sistema za podporo uporabnikom. Izvede testiranje uporabnosti za vse uporabniške vmesnike.
Testiranje	Oceni načrtovanje in testira značilnosti. Pripravi plan in urnik za testiranje značilnosti. Pripravi metode in merila za odkrivanje hroščev. Izdela strategijo testiranja.
Logistično upravljanje	Oceni načrtovanje s stališča namestitve, upravljanja, vzdrževanja in celotnih stroškov vzdrževanja. Izdela urnik in plan namestitve ter vzdrževanja.

[6]

3.2.3. Metode dela

V uporabi je proces načrtovanja MSF (MSF Design proces). Ta predstavlja učinkovito orodje, ki zagotavlja, da arhitektura aplikacije izraža potrebe uporabnikov in poslovnega procesa. Predvsem učinkovit je pri razvoju večnivojskih aplikacij, in je pravzaprav usmerjen k načrtovanju aplikacij takega tipa. Proces načrtovanja MSF je sestavljen iz treh različnih načinov načrtovanja: konceptualno, logično in fizično načrtovanje. Vsak način generira model z enakim imenom: model konceptualnega načrtovanja, model logičnega načrtovanja in model fizičnega načrtovanja. [2]

Tabela 3: Modeli načrtovanja

Načrtovanje	Perspektiva	Akcija
Konceptualno	Vidi problem s perspektive uporabnika in poslovnega procesa.	Definira problem in rešitev kot scenarije.
Logično	Vidi rešitev s perspektive projektnega tima.	Definira rešitev kot množico med seboj povezanih servisov.
Fizično	Vidi rešitev s perspektive razvijalcev.	Definira tehnologijo in servise rešitve.

Proces načrtovanja MSF vodi od otipljivih zahtev (primeri uporabe) do abstraktnega načrta aplikacije (konceptualno načrtovanje), nato do racionaliziranega načrta aplikacije (logično načrtovanje), do konkretnega načrta aplikacije (fizično načrtovanje) in končno do resničnega produkta.

Slika 3: Modeli načrtovanja

Tri vrste načrtovanja imajo naslednje lastnosti:

- prekrivanje (vsa tri načrtovanja lahko potekajo paralelno),
- iterativnost (vsako načrtovanje ima svoj cikel: preverjanje, testiranje načrtovanja in ponovno načrtovanje),
- spiralnost (vsaka posamezna iteracija vseh treh načrtovanj predstavlja napredek v smeri mejnika *potrditev projektnega plana*).

V vsakem trenutku morajo biti vsi trije načrti med seboj skladni. Če je bila izvedena sprememba na fizičnem načrtu, je potrebno ustrezno spremeniti tudi logični in konceptualni načrt in obratno. [2]

3.2.4. Konceptualno načrtovanje

Cilj: Identifikacija poslovnih potreb in razumevanje, kakšno je delo uporabnikov in kaj pri tem potrebujejo.
Izhod: Množica modelov z informacijami, primeri uporabe, scenariji primerov uporabe in dokumenti, ki predstavljajo trenutno in bodoče stanje.
Konceptualno načrtovanje generira scenarije, ki izražajo celotne in točne zahteve z vključenimi strankami, uporabniki in drugimi vpletenimi in predstavlja jezik uporabnikov. Začne se lahko že v *fazi vizije* in sicer takrat, ko so člani tima enotni v izjavah vizije. Konceptualno načrtovanje ne vsebuje pristopa oziroma tehnologije, potrebne za izgradnjo rešitve. Omejuje se samo na grobe skice in scenarije za izgradnjo rešitve. To so enostavno razumljivi modeli, pripravljeni s strani stranke, uporabnikov in načrtovalcev.
Konceptualno načrtovanje sestoji iz treh korakov: raziskava, analiza in racionalizacija. [1]

3.2.4.1. Raziskava

Pred začetkom procesa konceptualnega načrtovanja mora tim najprej določiti cilj njihovih raziskav. Tako naj se tim najprej omeji na opis glavnih procesov ustreznega poslovnega področja. To naj vključuje:

- opis bistvenih poslovnih procesov in njihovih robnih pogojev skupaj s funkcionalnimi elementi poslovanja,
- grob opis poslovnih transakcij znotraj teh poslovnih procesov in
- opis stranke in uporabnikov.

Za opis in prikaz teh aktivnosti se pri uporabi notacij UML uporabljajo diagrami uporabe, diagrami aktivnosti in diagrami sodelovanja.
Bistven poslovni proces:

- vodi poslovanje,

- direktno naslavlja strateške usmeritve in tekmovalnost,
- ima lastnike in stranke, ki jih lahko identificiramo,
- je definiran tako, da je razumljiv zunanjim strankam ali dobaviteljem kakor tudi osebju znotraj podjetja,
- je diskretno oziroma minimalno odvisen od ostalih bistvenih procesov

Bistvo raziskave je v pridobitvi celotne in točne slike, kjer je kvaliteta pridobljenih informacij pomembnejša od kvantitete pridobljenih podatkov.
Raziskavo je potrebno narediti tudi na uporabnikih oziroma skupini uporabnikov. Identificirati je potrebno čim več skupin uporabnikov, vključujoč lastnike organizacije, uporabnike znotraj organizacije in uporabnike zunaj organizacije, kot so dobavitelji in stranke. Za vsako skupino je potrebno izdelati svoj profil, ki pove, kakšna je njena vloga v organizaciji, njen oddelek, lokacija in njeno sodelovanje pri specifičnih aktivnostih. Te skupine in uporabnike pa je potrebno tudi povezati s predhodno zbranimi procesi in aktivnostmi. [2]

3.2.4.2. Analiza

Prva naloga drugega koraka, koraka analize, je potrditev rezultatov koraka raziskave, najpogosteje s skupinskim usklajevanjem. Ko je raziskava usklajena, je naslednja naloga priprava modelov, ki ponazarjajo kontekst, procesni tok in vrstni red nalog. Obstajata dva modela: diagrami primerov uporabe s scenariji primerov uporabe in diagrami aktivnosti. [2]

3.2.4.3. Racionalizacija

V tem koraku je bistvenega pomena določitev izboljšav, kjer je to možno. Kjer je mogoče, mora celoten tim, po možnosti tudi z zunanjo pomočjo, optimizirati poslovne procese. Kaj naj se optimizira? Cilj je v eliminaciji:

- neefektivnosti,
- neopisanih in nepotrebnih korakov,
- odvečnih in neefektivnih praks in procesov,
- nefunkcionalne politike in
- prevoza in zamudnega časa.

Tim si mora predstavljati in opisati bodoče stanje. Ko je bodoče stanje vizualizirano, se lahko na podlagi tega pripravi ustrezne nove scenarije.
Pri optimizaciji procesov je potrebno upoštevati naslednje osnove načrtovanja:

- kršitev pravil,
- upoštevanje ciljev učinkovitosti delovanja,
- načrtovanje na podlagi produktov in storitev,
- odstranitev birokratskih in drugih ovir,
- izboljšava produktivnosti,
- kje lahko tehnologija nudi podporo in
- drobljenje procesov na manjše procese.

Po izdelavi novih scenarijev je zadnji korak potrditev le teh, torej zagotovitev, da rešujejo poslovni problem. Tim doseže to z:

- izdelavo sistema preverjanja ustreznosti koncepta različice sistema,
- uporabo sistema preverjanja ustreznosti koncepta različice za predstavitev načrta uporabniškega vmesnika,

- pridobitvijo povratnih informacij o uporabnosti in procesu in s ponavljanjem, dokler stranka in uporabniki niso zadovoljni.

Sistem sistema preverjanja ustreznosti koncepta naj ima v zgodnjem načrtovanju samo osnovne značilnosti, načrt uporabniškega vmesnika in okvirno strukturo sistema. Zavzame lahko naslednje oblike:

- aplikacija, ki prikazujejo osnovno funkcionalnost,
- snemalne knjige (papirnate ali ekranske oblike),
- papirnate prototipe celotne strukture in uporabniške interakcije,
- MS PowerPoint dokument, ki prikazuje osnovne elemente sistema in demonstracijo navigacije skozi sistem za eno ali več opravil.

Ob napredovanju načrtovanja lahko prototipi dobijo definirano podobo, ki omogoča timu ocenitev vizualne oblike in nekaterih drugih načrtovalskih podrobnosti, predvsem uporabniškega vmesnika.
Pomembno je, da se ne poskuša preoblikovati celotnega poslovnega procesa z začetno različico aplikacije. To se lahko doseže v kasnejših scenarijih, ki nudijo dodatne izboljšave modeliranega poslovnega procesa. [2]

3.2.5. Logično načrtovanje

Cilj: Logično načrtovanje je proces opisovanja rešitve v organizacijskem, strukturnem in sintaktičnem smislu in sodelovanju njenih delov s perspektive projektnega tima. Predstavlja pogled na rešitev, kot jo vidi projektni tim.
Izhod: Izhodišče je množica poslovnih objektov z ustreznimi storitvami, atributi in povezavami, dizajn visokonivojskega uporabniškega vmesnika in dizajn logičnega podatkovnega modela.
Logično načrtovanje upravlja in odstranjuje kompleksnost z definiranjem rešitve, opisovanjem njenih delov in opisov, kako ti deli sodelujejo med seboj in tako rešijo problem. Odkriva in odstranjuje nekonsistenco konceptualnega modela. Odstranjuje redundantnost in identificira potencialno ponovno uporabo. Predstavlja tudi temelj za fizično načrtovanje. Razločno predstavi pogled na rešitev celotnemu timu.
Logično načrtovanje je neodvisno od fizične implementacije. Primarno se je potrebno osredotočiti na to, kaj naj sistem dela.
Logično načrtovanje vsebuje dva koraka: analiza in realizacija. [1]

3.2.5.1. Analiza

Cilj analize je pretvorba scenarijev, izdelanih v konceptualnem načrtovanju, za uporabo v logičnem načrtovanju. Moduli so bistveni primeri uporabe sistema, kakor tudi servisi ali akcije, ki se v njih izvajajo in njihove povezave. Modul je logična enota, ki abstraktno predstavlja primere uporabe in njihove scenarije. Tim mora za vsak modul identificirati njegove servise, objekte, atribute in povezave. [2]

3.2.5.2. Racionalizacija

Prva naloga racionalizacije je kreiranje do sedaj še ne kreiranih servisov in objektov. Kreiranje pa je potrebno bodisi zaradi predvidene potrebe ali zaradi potrebe po kontroli. [2]

3.2.6. Fizično načrtovanje

Cilj: Uporabiti želimo tehnologijo na že pripravljenem logičnem načrtu upoštevajoč elemente implementacije in učinkovitosti delovanja.

Izhod: Množica komponent, načrt uporabniškega vmesnika za določeno platformo in fizični podatkovni načrt.
Fizični načrt predstavlja osnovo funkcionalne specifikacije, ki jo *razvoj, testiranje in logistično upravljanje* uporabijo za zagotavljanje kvalitete. To je pomembno, kajti fizični načrt predstavlja zadnjo možnost preverjanja pred kodiranjem. Razvoj in dostava se lahko pričneta tudi že pred zamrznitvijo fizičnega načrta, kar omogoča naknadno čiščenje načrta, vendar pa mora biti zamrznjen pred stabilizacijo rešitve.
Fizični načrt ponuja osnovo drugim nalogam *faze planiranja*:

- izdelava približkov za ceno, terminski plan in plan virov,
- osveževanje analize tveganja.

Fizično načrtovanje sestoji iz štirih korakov: raziskava, analiza, racionalizacija in implementacija. [1]

3.2.6.1. Raziskava

Cilja koraka raziskave sta dvosmerna in sicer:

- določitev infrastrukturnih omejitev in fizičnih zahtev aplikacije in
- upravljanje s tveganjem konflikta med fizičnimi infrastrukturnimi omejitvami in fizičnimi zahtevami aplikacije.

Celoten tim pripravi spisek zahtev in ustreznih omejitev. Seznam je potrebno pripraviti z različnih perspektiv kot so:

- učinkovitosti delovanja,
- razmerje med ceno in dobičkom,
- dostava,
- vzdrževanje,
- izbira tehnologij,
- zanesljivost,
- dostopnost in
- varnost.

Raziskava trenutne fizične infrastrukture poteka z uporabo topologij, predstavljenih kot načrti, ki predstavljajo različne aspekte infrastrukture. Za potrebe planiranja je smiselna priprava omrežnih, podatkovnih in komponentnih topologij, ki predstavljajo trenutno stanje infrastrukture. [2]

3.2.6.2. Analiza

Poglavitna naloga koraka analize je izbira kandidatov tehnologije za implementacijo na podlagi razumevanja zahtev aplikacije. To so torej kandidati, odločitev je izvedena kasneje. Ko so potencialne tehnologije izbrane, lahko tim pripravi predhodni model dostave.
Pri ocenjevanju tehnologij naj tim na prvem mestu upošteva poslovne zahteve, šele nato zahteve arhitekture podjetja ter na koncu še tehnološke zahteve.

Glede na poslovne zahteve je potrebno upoštevati:

- sposobnost, ali bo tehnologija sploh zadovoljevala poslovne zahteve,
- ceno produkta (potrebno se je zavedati celotne cene: cene razvijalcev, strežnikov, licenc, nadgradenj),
- izkušnje (potrebno se je zavedati, da imajo lahko izkušnje ali pa njihovo pomanjkanje velik vpliv, katere izkušnje so na voljo v obliki tečajev),

- zrelost ali inovacijo,
- vzdrževanje (podpora je potrebna tako tehnologiji, kakor tudi rešitvi, izdelani s pomočjo te tehnologije, možnosti podpore so prodajalec, zunanja oskrba in informacijska deska).

V razmislek pridejo še dostava, konkurenčna prednost, čas trženja in zaznavanje industrije.

Glede na arhitekturo podjetja je potrebno upoštevati:

- **usklajenost z arhitekturnimi cilji podjetja**
  Aplikacija naj ustreza zastavljenim arhitekturnim ciljem podjetja. Cilji bazirajo na štirih principih in modelih arhitekture podjetja: poslovanje, aplikacija, informacija in tehnologija.
- **vdanost arhitekturi podjetja**
  Arhitektura podjetja bo določala tako plane trenutnega kakor tudi bodočega stanja.
- **možnost razširjanja**
  Skalabilnost aplikacije je lahko upoštevana znotraj širjenja poslovanja.
- **večuporabnost**
  Aplikacija mora sodelovati tudi z drugimi aplikacijami znotraj organizacije.

Glede na tehnologijo je potrebno upoštevati:

- **programski jezik**
  Pri izbiri programskega jezika za razvoj komponent je potrebno za različne naloge razmisliti o različnih jezikih.
- **standarde za komunikacijo med komponentami**
  Pri izbiri standardov za komunikacijo med komponentami je potrebno upoštevati integracijo med platformami z vidika zmogljivosti in učinkovitosti delovanja.
- **metode za dostop do podatkov**
  Upoštevati je potrebno predvsem učinkovitosti delovanja, standarde in bodoče usmeritve kakor tudi raznolikost podatkovnih baz in upravljanja z njimi.
- **sistemske storitve**
- **operacijski sistem**
  Pri izbiri operacijskega sistema lahko storitve, ki jih le ta ponuja, znatno znižajo potrebe po kodiranju aplikacije. Poleg tega lahko le ta poseduje ustrezne varnostne mehanizme za skalabilnost.

Misleč na kandidate za tehnologijo lahko sedaj tim izdela predhodni model dostave, ki je sestavljen iz omrežne, podatkovne in komponente topologije. Na tej točki so to šele predlagane in ne izbrane topologije.

Omrežna topologije predstavlja infrastrukturni načrt, ki ponazarja lokacije strojne opreme in medsebojne povezave.

Podatkovna topologija je predstavljena z načrtom distribucije podatkov, ki ponazarja lokacije podatkovnih baz v povezavi z omrežno topologijo.

Komponenta topologija je predstavljena z načrtom distribucije komponent, ki ponazarja lokacije komponent in njihovih servisov v povezavi z omrežno topologijo.

Vsi načrti ponazarjajo trenutno stanje in so bili razviti že v prejšnjem koraku analize, tu pa so mogoče potrebne izboljšave, ki podpirajo novi fizični načrt. [1]

3.2.6.3. Racionalizacija

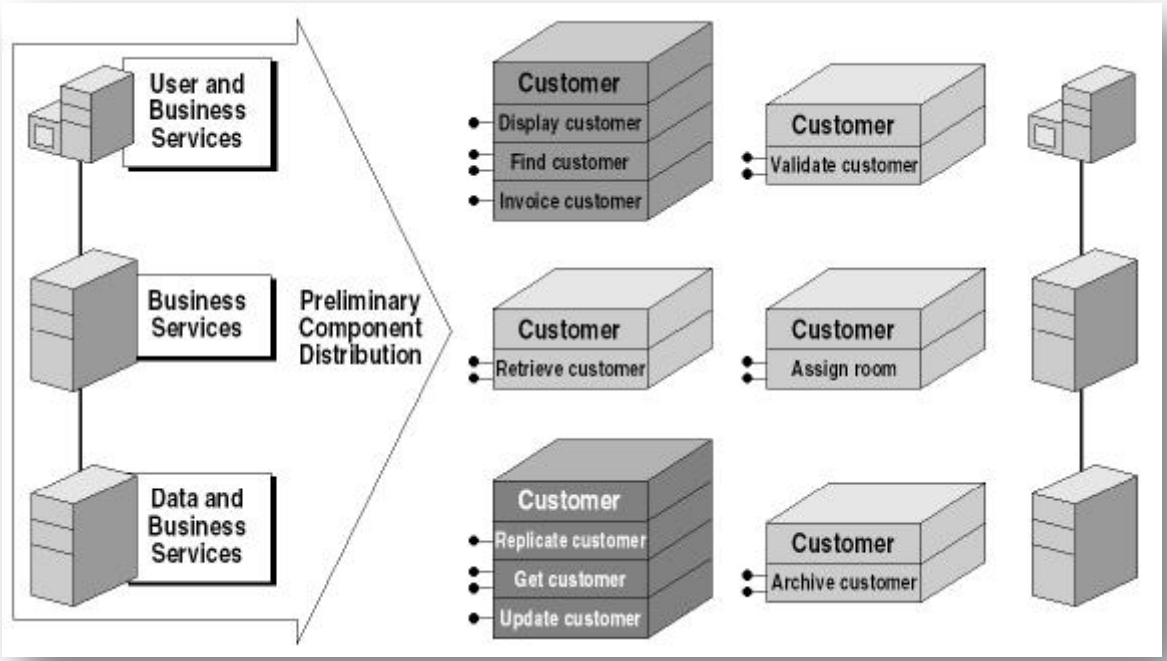
Raziskava in analiza sta končani in čas je za dokončanje fizičnega načrtovanja in sicer določitev pakiranja in distribucije.

Pri določanju strategije pakiranja in distribucije naj tim sledi naslednjim enostavnim korakom. Najprej je potrebno razmisliti o različnih načelih pakiranja ali razlogih za izbor določene strategije. Med te spadajo naslednji:

- kategorija storitve,
- skalabilnost,
- učinkovitost delovanja,
- upravljivost,
- ponovna uporaba,
- poslovni kontekst in
- razdrobljenost.

Drugi korak je v uskladitvi strategije s programskim modelom.
Tretji korak je določitev načrtovalskih izmenjav, ki vplivajo na strategijo.

Tim je sedaj pripravljen na začetek definiranja komponent. Čeprav za to obstaja več načinov, je priporočljivo, da je osnovna ideja izdelana na podlagi aplikacijskega modela MSF. Ta pa predstavlja tri nivoje storitev: uporabniški, poslovni in podatkovni nivo. Storitve so porazdeljeno glede na topologijo, kakor kaže slika. Tim lahko nato pakira kandidatke za komponente iz iste logične storitve na isto mesto v topologiji.



Slika 4: Trije nivoji storitev

Ko je komponenti model zaključen, lahko tim zaključi tudi porazdelitev teh komponent po nivojih. Ni pa nujno, da uporaba treh logičnih nivojev ne pomeni porazdelitev le teh na tri fizične lokacije. Povsem možna je rešitev, kjer N-nivojska aplikacija teče na enem

računalniku. Prav tako pa lahko distribucijska strategija za kompleksno aplikacijo pomeni porazdelitev na deset, dvajset ali več fizičnih lokacij.

Med racionalizacijo naj tim preverja komponente in njihovo pakiranje ter dostavo. To naj izvršita predvsem vlogi: *razvoj* in *testiranje*. Preverjajo naj, ali:

- načrtovane komponente vsebujejo vse storitve načrtovanja v logičnem načrtu,
- so bile upoštevane vse odvisnosti komponent/storitev,
- porazdelitev komponent glede na topologijo ustreza načrtovani,
- so storitve grupirane po komponentah tako, da so v ravnovesju z fizično lokacijo, pakiranjem in omejitvami tehnologije,
- komponentni plan upošteva razširljivost (navzgor kakor tudi navzdol). [2]

3.2.6.4. Implementacija

Zadnji korak fizičnega načrtovanja je implementacija. V tem koraku tim specificira programski model, ki ga nato *razvoj* uporabi pri razvoju. Tu so specificirani tudi vsi vmesniki komponent in njihova interna struktura.

Programski model, ki mu lahko rečemo tudi programska specifikacija ali standard, zagotavlja naslednje:

- predpisuje, kako uporabljati izbrane tehnologije,
- določa dizajnerske smernice pri specifikaciji komponent, kar pripomore k večji konsistenci celotnega projekta,
- uporablja različne pristope k reševanju različnih aspektov rešitve.

Pri izdelavi programskega modela je potrebno upoštevati več aspektov kot so:

- **tehnologija implementacije**
  Programski jezik, API, strežniki in strežniške tehnologije, …
- **objekti z ali brez stanja**
  Objekt s stanjem hrani stanje, pridobljeno s strani več odjemalcev, medtem ko objekt brez stanje ne hrani ničesar.
- **klici funkcij v-proces proti iz-procesa**
  Ali naj se funkcija izvaja v svojem ali v kličočem procesu?
- **povezani proti nepovezanim modelom (sinhronski proti asinhronskim programskim modelom)**
  Sinhronski programski model blokira nadaljevanje izvajanja kličoče komponente, dokler klicani vmesnik ne zaključi z delom. Asinhronski model pa dopušča komponentam pošiljanje sporočil drugim komponentam in nadaljevanje izvajanja.
- **nitni model**
  Izbira nitnega modela je zelo težka, ker je odvisna od funkcije, ki jo komponenta izvaja.
- **upravljanje napak**
- **varnost**
  Vsaka komponenta ima štiri možnosti zaščite: na podlagi komponente bodisi na nivoju metode, nivoju vmesnika ali pa na nivoju komponente; na podlagi podatkovne baze; na podlagi uporabnika bodisi sistemsko ali pa programsko realizirano; na podlagi vlog.
- **dostava**
  Pri uporabi treh logičnih nivojev ne pomeni tudi treh fizičnih. Logični nivoji bodo razporejeni po fizičnih nivojih.

Ko je programski model začrtan, je tim pripravljen na specifikacijo zunanje strukture komponent. Ta pa je predstavljena s pomočjo vmesnikov:

- je dogovor, ki predstavlja povezavo ponudnik-odjemalec med komponentami,
- je način za dostop do vsebovanih storitev,
- predstavlja eno ali več storitev,
- vsebuje atribute vsebovanih objektov.

Specifikacija vmesnika komponente mora vsebovati vse možne načine dostopa do komponente, po možnosti s pripadajočimi primeri za vsak način.

Pri izdelavi vmesnikov komponent je potrebno upoštevati naslednja dejstva:

- izdelan vmesnik predstavlja dogovor, ki naj bi bil trajen,
- sprememba obstoječega vmesnika naj bi bila izdelana kot nova komponenta ali nov vmesnik,
- podatkovni tipi izdelanih atributov morajo biti podprti s strani storitvenega vmesnika odjemalca.

Vmesnik predstavlja dogovor, ki določa parametre, podatkovne tipe, sodelovalne standarde in opise samega vmesnika. Stopnja specificiranosti je odvisna od uporabniških zahtev in končno od zahtev ponovne uporabe.

Končno je tim pripravljen za specificiranje notranje strukture komponent. Pri definiranju notranje strukture komponente je potrebno upoštevati več faktorjev. Najpomembnejši je gotovo programski jezik ali orodje za implementacijo komponente.
Ko je notranja struktura komponent specificirana, so osnove za implementacijo vzpostavljene. [2]

3.2.7. Mejnik odobritev projektnega plana in izdelki faze planiranja

Cilj faze *planiranja* je doseg mejnika *odobritev projektnega plana*, ki predstavlja glavni cilj tega koraka in dogovor oziroma pogodbo med stranko, naročnikom in timom.
Za doseg mejnika *odobritev vizije* so potrebni naslednji izdelki:

- funkcionalna specifikacija,
- glavni projektni plan in urnik in
- revidiran dokument tveganj.

Na tem mejniku se tim še vedno lahko odloči za opustitev ali nadaljevanje projekta. Kajti dobro zaključena *faza planiranja* lahko timu in stranki oziroma naročniku zagotovi nadaljevanje projekta z zaupanjem in prepričanjem, da je bilo storjeno vse za uspešnost projekta. [2]

3.2.7.1. Funkcionalna specifikacija

Funkcionalna specifikacija je glavni izdelek *faze planiranja*. Predstavlja podrobno zbirko specifikacij za projekt in služi kot pogodba med stranko in projektnim timom. Pretirano je reči, da se vse dosedanje delo manifestira v funkcionalni specifikaciji in da bo nadaljnje delo izhajalo iz nje.
Lastnik funkcionalne specifikacije je programsko upravljanje, ki je tudi zadolženo za dokončanje le te. To pa ne pomeni, da mora programsko upravljanje napisati celotno funkcionalno specifikacijo, ampak mora odsevati razumevanje produkta celotnega tima.

Vsebina funkcionalne specifikacije

Funkcionalna specifikacija naj vsebuje vse spodaj naštete elemente:

Tabela 4: Funkcionalne specifikacije

Element	Opis
Povzetek vizije	Kaj tim želi, da bi produkt bil, njen zagovor in ključne omejitve.
Cilji načrtovanja	Kaj želi tim s projektom doseči. Razvoj uporabi te cilje kot izhodišče pri odločitvah o zmogljivosti, stabilnosti, uporabnosti in dostopnosti.
Zahteve	Kaj stranka, uporabniki in naročnik mislijo, da bo projekt. Te zahteve naj bodo razvrščene po prioritetah.
Povzetek uporabljivosti	Kdaj bo produkt uporabljen in kdo ga bo uporabljal.
Značilnosti	Kaj točno je produkt dela. Značilnosti produkta, razvrščene po prioritetah.
Odvisnosti	Od česa je produkt odvisen. Spisek zunanjih objektov, od katerih bo produkt odvisen.
Povzetek urnika	Opis urnika. Povzetek projektnega urnika z poudarkom na vmesne mejnike, vmesne produkte in datum zaključka.
Tveganja	Kakšna so tveganja.
Dodatki	Vse ostalo. Množica izdelkov procesa načrtovanja, ki jih je tim uporabil pri pripravi funkcionalne specifikacije.

Pregled in sprejetje funkcionalne specifikacije

Glavni cilj revizije je pridobiti mnenja vseh vlog. Končni cilj priprave funkcionalne specifikacije pa je njeno sprejetje oziroma konsenz vseh vlog. Projektni tim, stranka oziroma naročnik naj potrdijo, da funkcionalna specifikacija točno in pravilno dokumentira pričakovanja projekta. [2]

Strinjanja vlog pri funkcionalni specifikaciji:

Tabela 5: Vloge in njihove zadolžitve

Vloga	Zadolžitve
Stranka	Produkt zadostuje poslovnim potrebam. Stranka je pripravljena sprejeti plan in vire za razvoj za obseg, predstavljen v funkcionalni specifikaciji.
Produktno upravljanje	Produkt zadostuje znane zahteve. Produktno upravljanje verjame, da funkcionalna specifikacija odseva produkt, ki bo zadostoval znanim zahtevam, ko bo izdelan.
Projektno upravljanje	Zadolžitve in plani tima so realistični. Projektno upravljanje verjame, da je jasno, kdo je zadolžen za posamezno funkcijo in da je sprejeti plan realističen.
Razvoj	Produkt je možno razviti. Razvoj je uspešno spremljal tveganja razvoja in verjame, da so tveganja za dosego zastavljenega datuma dokončanja realistična.
Uvajanje	Produkt je uporaben in potrebe po podpori uporabniku so definirane. Uvajanje je seznanjeno, kdo so uporabniki in kako bo produkt uporabljen in podprt.
Testiranje	Produkt je možno testirati in stabilizirati. Testiranje ima definirano strategijo za testiranje vseh aspektov funkcionalne specifikacije.
Logistično upravljanje	Produkt je možno vzdrževati in namestiti. Logistično upravljanje je seznanjeno z organizacijskimi, aplikacijskimi in sistemskimi izhodišči in jamči namestitev produkta.
Vse vloge	Vsi se strinjajo z datumom zaključka.

3.2.7.2. Projektni plan

Glavni projektni plan nam pove, kako bo produkt izdelan z združevanjem detajlnih planov članov projektnega tima. Namen glavnega projektnega plana je:

- omogočanje uskladitev tima in plana dela za določeno vlogo,
- opisovanje, kako bodo tim in vloge izvrševale svoje vloge,
- omogočanje sinhronizacije planov celotnega tima.

Lastnik glavnega projektnega plana je projektno upravljanje, ker predstavlja glavnega koordinatorja planiranja in dejanskega dela na projektu. Vseeno pa je vsaka vloga zadolžena za izdelavo in vzdrževanje lastnega realnega projektnega plana glede na glavni plan. [2]

Glavni projektni plan naj vsebuje naslednje spodaj naštete elemente:

Tabela 6: Vsebina projektnega plana

Element	Opis
Razvoj	Opisuje, kako bo razvoj izdelal vse, kar je zapisano v funkcionalni specifikaciji. Opisuje stvari kot so orodja, metodologije, primeri uporabe, zaporedja dogodkov in metode za testiranje.
Testiranje	Vsebuje strategijo testiranja: specifična področja, ki morajo biti testirana ter ustrezne vire (oprema, ljudje).
Izobraževanje	Vsebuje strategijo in plan za izdelavo vsega potrebnega testnega materiala.
Podpora uporabnikom	Vsebuje strategijo in podrobnosti za razvoj vsega, kar bodo uporabniki potrebovali za lažje delo (čarovniki, priročnik za uporabo, …).
Komunikacija	Vsebuje marketinško strategijo in aktivnosti promocije uporabnikom.
Namestitev	Vsebuje strategijo in podroben plan za pripravo končnih uporabnikov in administratorjev pred in med namestitvijo.

[2]

3.2.7.3. Revidirani dokument tveganj

Ob koncu faze *planiranja* mora tim imeti precej jasno predstavo o tveganjih, povezanih s projektom ter njihovimi posledicami ter njihovo prioriteto.
Projektno upravljanje si dokument lasti in je hkrati odgovorno da je revizija končana in točna.
Revidirani dokument tveganj vsebuje zbirko vseh tveganj različnih članov tima.
Dokument je v pomoč pri usklajevanju seznama tveganj celotnega tima. [1]

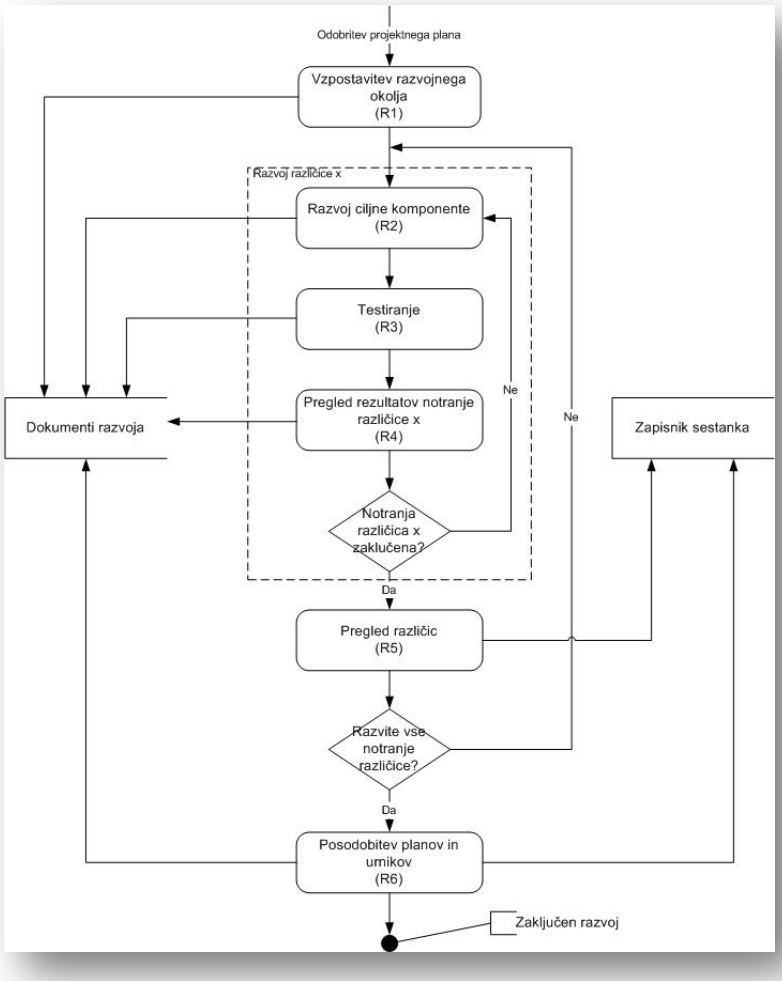
3.3. Razvoj

3.3.1. Splošno

V *fazi razvoja* si zastavimo vprašanje »Kako uspešno izvesti razvojni proces, da dobimo pravi produkt ob pravem času?«. Odgovor najdemo v skupnemu razumevanju vizije produkta. Da bi bila *faza razvoja* še bolj uspešna, mora tim poleg razumevanja vizije upoštevati tudi realne omejitve razvoja delujočega produkta. Po zaključku faze razvoja in dosegu mejnika imamo zaključen razvoj z naslednjimi lastnostmi:

- implementirane so vse značilnosti produkta, čeprav ne v celoti optimizirane,
- izvedeno je bilo osnovno testiranje in pripravljen je spisek tekočih hroščev (ni nujno, da so že odpravljeni),
- člani tima in naročnik se strinjajo, da vključene značilnosti ustrezajo viziji in planiranju in so uspešno implementirane,
- za testiranje in stabilizacijo so pripravljeni okvirni materiali uporabniških učinkovitosti delovanja.

Pri prehodu iz planiranja v razvoj tim ne more razviti vseh funkcionalnosti hkrati, temveč mora biti kodiranje razdeljeno v več različic (segmentov). Le te je potrebno ustrezno upravljati in nadzirati in na koncu združiti v končni produkt.
Razvoj predstavlja približno 35 do 40% celotnega projekta. [2]



Slika 5: Shema razvoja

3.3.2. Kdo ga izvede?

Spodnja tabela opisuje tipične zadolžitve vlog v *fazi razvoja*. Vodja vsake vloge skrbi za izvršitev zadanih nalog in komunikacijo z ostalimi člani tima.

Tabela 7: Tipične zadolžitve v fazi razvoja

Vloga	Zadolžitve
Produktno upravljanje	Zadolženo je za upravljanje z zahtevami in pričakovanji stranke. Prav tako skrbijo za informiranje stranke in zunanjih naročnikov. Zadolženo pa je tudi za pripravo na alfa in beta različico.
Projektno upravljanje	Vodi komunikacijo preko celotnega projektnega tima in koordinira interne različice prek celotnega tima. Prav tako je zadolženo za možno revizijo dokumentov, pripravljenih v *fazi planiranja*.
Razvoj	Izdela vso kodo, potrebno za implementacijo vseh značilnosti produkta. Prav tako je zadolženo za osnovno testiranje kode in značilnosti produkta.
Uvajanje	Izvede osnovno testiranje uporabnosti produkta in prične s testiranjem uporabniških učinkovitosti delovanja. Koordinira uporabniški forum za alfa in beta različici produkta. Skrbi za izdelavo uporabniških priročnikov in dokumentacije za vzdrževanje.
Testiranje	Izdela podroben plan testiranja (scenariji, podatki, skripti,…) za izvedbo osnovnega funkcionalnega testiranja. Le to je izvedeno v *fazi razvoja* za potrditev internih različic. Osnovna naloga testiranja je identificiranje in sledenje hroščem ter vodenje celotne dokumentacije testiranja.
Logistično upravljanje	Nudi logistično podporo pri razvoju. Pripravi tudi ves material in dokumentacijo, potrebno za namestitev in osnovno vzdrževanje alfa in beta različic produkta.

[6]

3.3.3. Metode dela

Proces razvoja MSF je sestavljen iz treh korakov: *analiza* in *racionalizacija*, *implementacija* in *validacija*. [2]

3.3.3.1. Analiza in Racionalizacija

V tej *fazi razvoja* ima tim konkreten plan akcij z nekaj spremenljivkami. V tej fazi sicer dodatne analize niso potrebne, a naj se faza kljub temu prične z analizo trenutnega načrta aplikacije.

Analizirati je potrebno projektni plan in urnik ter identificirati vire, ki jih je potrebno dodeliti za kodiranje. Po pregledu skupkov značilnosti, pripravljenih v *fazi planiranja*, razvojni tim razdeli razvoj posameznih značilnosti članom. Tu je potrebno upoštevati tudi delitev aplikacije v servisne plasti.

Tim za testiranje analizira načrt produkta in določi, kdo bo izvedel posamezen test in pregleda primere uporabe, da zagotovi ustrezno testiranje uporabnosti produkta. Pripravi tudi izčrpno zbirko testnih scenarijev na podlagi primerov uporabe, fizičnega načrta in nefunkcionalnih zahtev, kot so aplikacijske in uporabniške zahteve po učinkovitosti delovanja.

Po izdelavi posamezne interne različice naj tim analizira odziv uporabnikov in tima za testiranje in vzdrževanje, s čemer se lahko določi uspešnost trenutne interne različice produkta. Vsi hrošči in drugi problemi produkta morajo biti najprej znani in nato ustrezno odpravljeni. [2]

3.3.4. Implementacija

Glavna naloga je seveda implementacija izvorne kode produkta, ki jo pripravi razvojni tim. Poleg tega pa mora uvajalni tim pripraviti dokumente, potrebne za vzdrževanje aplikacije kakor tudi uporabnikov. Poleg tradicionalnih uporabniških in namestitvenih dokumentov se lahko tim osredotoči tudi na on-line pomoč ter razne vodiče ter programske čarovnike za izboljšavo uporabnosti in podpore. [2]

3.3.4.1. Validacija

Validacija je delo celotnega tima. Posebno pa je to odgovornost razvojnega tima in tima za testiranje. Validacija poteka na področjih uporabnosti, preformans, sledenja hroščem in ideji o produktu brez napake.

Zaradi več internih različic v *fazi razvoja* je potrebno v validacijo vključiti kar precejšen del tima. Za večjo učinkovitost pri validaciji kode naj razvojni tim in tim za testiranje v čim večji meri avtomatizirata proces testiranja. [2]

3.3.5. Mejnik zaključen razvoj ter izdelki faze razvoja

Glavni cilj *faze razvoja* je doseg mejnika *zaključen razvoj*. Za doseg mejnika *zaključen razvoj* so potrebni naslednji izdelki:

- **revidirana funkcionalna specifikacija**
  Prikazuje dejansko stanje izdelane aplikacije z vsemi spremembami v *fazi razvoja*.
- **revidirani projektni plan in urnik**
  Osvežen plan in urnik, ki predstavlja dejanski plan in urnik implementacije.
- **revidirani dokument tveganj**
  Dodana so še podrobna tveganja posredovana od posameznih članov tima. [2]

3.3.5.1. Interne različice produkta

Interne različice sledijo celotnemu konceptu razvoja v različicah. Obe vključujeta inkrementalno dodajanje funkcionalnosti na že zastavljeno osnovo. Interne različice omogočajo timu stabilizacijo produkta, dosego ciljev kvalitete in vajo za oddajo produkta. Pri uporabi internih različic v *fazi razvoja* je pomembno, da se določi znana osnova, ki se kasneje uporabi kot primerjalno orodje.

Interne različice nastajajo skozi celotno *fazo razvoja*. Vsaka različica je načrtovana tako, da razširja oziroma dopolnjuje prejšnjo, kakor kaže spodnja slika. Neodvisnost različic nudi pogled na različico kakor že na končen produkt, kar še dodatno spodbudi tim v *fazi razvoja*. Revizija interne različice je pomemben del pri usmerjanju tima v učečo se organizacijo, ki uporablja najboljša praksa in se uči iz lastnih napak.

Slika 6: Različice produkta

Vsaki interni različici določimo nivo kakovosti, ki ga mora zadovoljiti, da je dosežen interni mejnik. Nivo kakovosti predstavlja merilno orodje, ki ga tim jasno določi. Vsaka interna različica ima v manjši meri tudi *fazo stabilizacije*, s pomočjo katere je dosežena določena kvaliteta.
Pri razvoju produkta tim s pomočjo zaporedoma sledečih internih različic inkrementalno dodaja skupke značilnosti, vse dokler produkt ni končan. Uspešnost delitve produkta v interne različice oziroma skupke značilnosti je odvisna predvsem od pravilnega planiranja v predhodni *fazi planiranja*. [2]

3.3.5.2. Izvorna koda in izvršilne datoteke

Izvorna koda predstavlja osnovo za končni produkt in je podvržena nadzoru kvalitete. Kvaliteti kode se ne smemo odreči, pa čeprav smo pod pritiskom časa. Upravljanje je zadolženo za izbiro kompromisa med kvaliteto kode in dosego zastavljenega roka. Da bi zmanjšali možnosti žrtvovanja kvalitete v želji po prihranku časa, mora tim postaviti standarde, ki:

- silijo razvijalce, da uporabljajo metodološki in disciplinirani pristop k kodiranju, ne glede na željo po uporabi bližnjic,
- konstantno opominjajo razvijalce, da je interna kvaliteta kode pomembna,

Standardi kodiranja vsebujejo naslednje elemente:

- poimenovanja,
- obliko (poravnave, zamiki, …),
- komentarje in
- kako kodirati in kako ne. [2]

3.3.5.3. Dokumenti podpore

Dokumenti podpore obsegajo vse od čarovnikov znotraj produkta pa do uporabniške dokumentacije in navodil ter materiala za tečaj. Kakor je zapisal Steve Maguire v knjigi Debugging the Development Process (Microsoft Press, 1994):

> »*Programerji morajo programirati z mislijo na uporabnika. Če programer misli, da je določena naloga neintuitivna ali počasna, je velika verjetnost, da bo enakega mnenja tudi uporabnik.*« [2]

3.3.5.4. Elementi testiranja

Združujejo različna orodja, ki jih lahko uporabimo pri načrtovanju in razvoju solidnega produkta.

Integrirano testiranje ob mejniku

Postopek testiranja ni omejen samo na *fazo stabilizacije*, saj je bistveni del *faze razvoja*. Specifikacija testiranja je zaključena ob mejniku *zaključen razvoj*, ko je število značilnosti fiksno. Ob koncu *faze razvoja* je produkt v alfa obliki (to nima povezave z alfa in beta testiranjem v *fazi razvoja*). Pri prehodu iz *faze razvoja* v *fazo stabilizacije* prehajamo iz testiranja obsega v testiranje uporabnosti. Primeri testiranja obsega na tipičnem projektu:

- testi celote,
- funkcionalni testi,
- testi prevoda,
- testi nazadovanja (mogoče deluje kaj slabše kakor je prej).

Tipični testi uporabnosti:

- test konfiguracije (produkt dela na ciljni HW in SW opremi),

- test združljivosti (interakcija z drugimi programi),
- test učinkovitosti delovanja in
- test dokumentacije in uporabniške pomoči (pregled napak).

Alfa test predstavlja prvi test celotnega produkta s pomočjo zunanjega vira. Beta testi so testi, izvedeni na demo produktu s pomočjo množice zunanjih virov s ciljem ugotoviti napake, ki jih projektni tim ni našel.
Pri odpravi hroščev je potrebna njihova klasifikacija. Osnovna elementa klasifikacije hroščev sta resnost in prioriteta. Resnost predstavlja vpliv hrošča na celoten produkt, če le ta ni odpravljen. Prioriteta pa je le merilo pomembnosti hrošča na stabilnost produkta, ki ga določi tim. Tipični nivoji klasifikacije resnosti:

- resnost 1: odpoved sistema ali nevarnost izgube podatkov,
- resnost 2: velik problem, hrošč predstavlja resno odstopanje funkcionalnosti,
- resnost 3: manjši problem, hrošč predstavlja manjše odstopanje funkcionalnosti,
- resnost 4: trivialno, v glavnem kozmetični problem.

Tipični nivoji klasifikacije prioritete:

- prioriteta 1: najvišja prioriteta, produkta ni mogoče zaključiti,
- prioriteta 2: visoka prioriteta, produkta ni mogoče zaključiti, vendar tim lahko doseže naslednjo interno različico,
- prioriteta 3: majhna prioriteta, hrošče se odpravi, če je dovolj časa,
- prioriteta 4: najnižja prioriteta, z odpravo teh hroščev se dosežejo izboljšave sistema.

Tipično produkta ni možno zaključiti, če je nivo resnosti in prioritete 1.
Odprava hrošča je notranji korak v smeri zaključka, ki je dosežen po potrditvi testerja, da odprava hrošča ni povzročila drugih težav.
Hrošči so tipično odpravljeni s statusi:

- **odpravljen**
  Razvoj hrošča odpravi, testira popravek, kodo, določi popravek končni različici, vrne poročilo nazaj osebi, ki je hrošč sporočila.
- **podvojen**
  Ponovljen hrošč že evidentiranega hrošča, vzpostavi se povezava z originalom.
- **odložen**
  Hrošč ne bo odpravljen v trenutni različici.
- **po načrtu**
  Obnašanje hrošča je namensko in je del funkcionalne specifikacije.
- **brez ponovitve**
  Razvoj ne more preveriti ponoviti pojavitve hrošča.
- **brez popravka**
  Hrošč ne bo odpravljen, ker tim odloči, da ni vreden truda. [2]

Dnevni prevod
Čeprav je lahko težaven za izvedbo, nudi velike koristi. Predstavlja enostaven prevod celotne kode aplikacije v izvršilno datoteko. Čeprav se imenuje dnevni, pa je uporaben tudi, če ga izvajamo na vsake tri ali štiri dni. Moč dnevnega prevoda predstavlja soočenje celotnega tima z napredkom posamezne ekipe, kar pomeni dodatno stimulacijo članom. Prav tako se že takoj vidijo težave integracije, ki se jih tako zaveda celoten tim. [2]

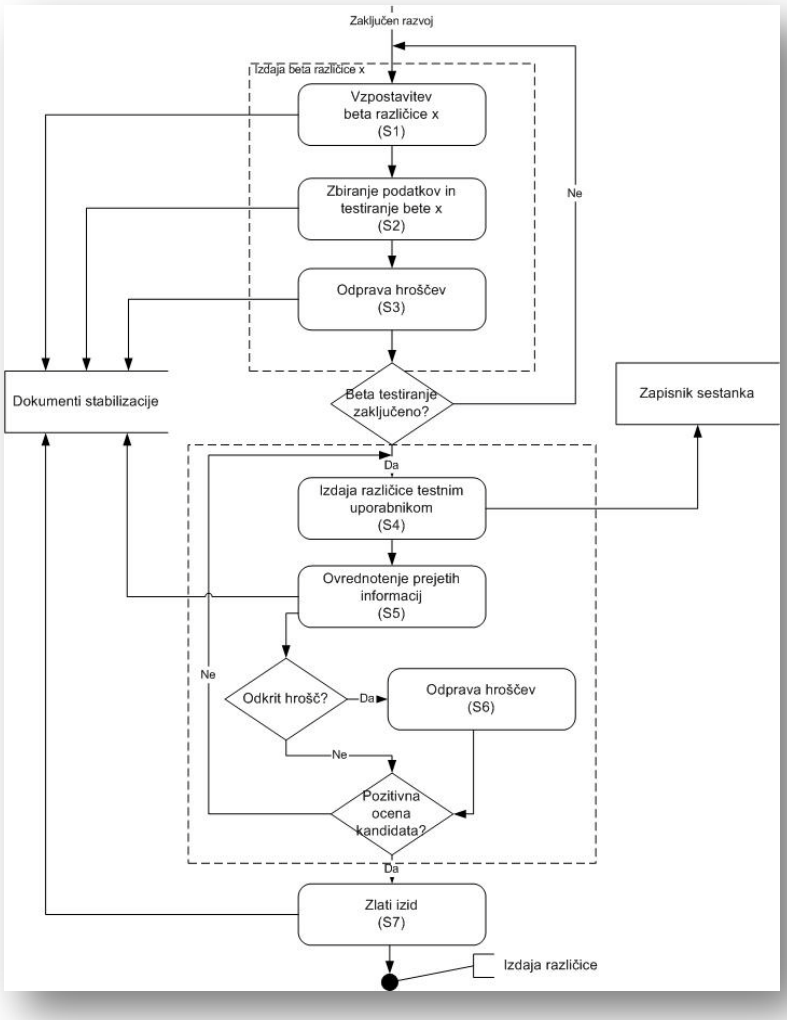
3.4. Stabilizacija

3.4.1. Splošno

Ko tim zaključi s kodiranjem izvorne kode in doseže mejnik *zaključen razvoj*, je naloga stabilizacije, da produkt dostavi stranki. V *fazi stabilizacije* korakov, podobnih prejšnjim fazam, ni. Namesto tega v *fazi stabilizacije* tim izda več različic produkta oziroma mora doseči več internih mejnikov vse do končnega produkta. Ti mejniki so odprava hroščev, uskladitev vseh izdelkov produkta, izdaja različice in izdatno testiranje izdane različice. Pri teh vmesnih oziroma mejnih različicah, ki vse ponavljajo enake korake, se tim omeji predvsem na testiranje s ciljem ugotavljanja hroščev in njihovo odpravo. Pomemben interni mejnik, ki naznanja bližanje končnemu produktu, je različica brez hrošča (RBH ali ZBR). To je prvi interni mejnik, kjer so bili vsi aktivni hrošči obravnavani (odpravljeni, podvojen, odložen, …).

Faza stabilizacije tudi pomeni dokončanje vseh dokumentov podpore. Ti morajo prav tako biti testirani in zaključeni, preden je izdana končna različica. Šele, ko je pripravljena vsa koda aplikacije kakor tudi ostali dokumenti, tim doseže cilj izdelave pravega produkta ob pravem času.

Stabilizacija predstavlja približno 10 do 15% celotnega projekta. [2]



Slika 7: Shema stabilizacije

3.4.2. Kdo jo izvede?

Spodnja tabela opisuje tipične zadolžitve vlog v *fazi stabilizacije*. Vodja vsake vloge skrbi za izvršitev zadanih nalog (glavna naloga je izdaja različice) in komunikacijo z ostalim člani tima.

Tabela 8: Tipične zadolžitve vlog v fazi stabilizacije

Vloga	Zadolžitve
Produktno upravljanje	Koordinira izdaje internih različic s stranko. Planira splavitev produkta.
Projektno upravljanje	Upravlja z beta in pilotskimi različicami produkta. Vzdržuje projektni urnik.
Razvoj	Omejeno na iskanje in odpravo hroščev. Zagotavlja, da je integracija produkta končana in uspešno testirana.
Uvajanje	Zagotovi dokončanje materialov za podporo uporabnikom. Določa in koordinira predavatelje in uporabnike za izobraževanje internih različic ter končnega produkta.
Testiranje	Izvede testiranje po planu. Najde, zabeleži in klasificira hrošče. Zaključi hrošče in zagotovi, da so ustrezno odpravljeni. Usmeri se tudi na testiranje uporabnosti, namestitve in konfiguracije.
Logistično upravljanje	Skrbi za namestitev, konfiguracijo, dostavo in podporo internih različic. Planira dostavo končnega produkta. Nudi podporo sistemski skupini in skupini za podporo ter skupini pomoč uporabnikom.

[6]

3.4.3. Metode dela

Kakor je bilo že omenjeno, *faza stabilizacije* ne poteka v korakih, podobnih prejšnjim fazam. Faza stabilizacije poteka preko internih mejnikov, ki jih mora tim doseči. Vsak interni mejnik predstavlja svojo različico produkta. Te interne različice si sledijo zaporedoma vse do končnega produkta. Pri vsakem od omenjenih internih mejnikov tim integrira in uskladi vse izdelke produkta. Z vsako interno različico je produkt testiran, odpravljeni so hrošči in izvedeni so popravki. Izdaja končnega produkta je skupna odločitev vseh vodij vlog, stranke ter sistemske skupine in skupine za podporo. Interni mejniki znotraj faze stabilizacije: *zmanjšanje hroščev, različica brez hroščev, kandidat za izdajo* in *končna izdaja produkta.* [2]

3.4.3.1. Iskanje hroščev

Tim lahko izda več internih različic, ki jih da v testiranje skupini uporabnikov in s tem omogoči še dodatno testiranje že v fazi razvoja. V fazi stabilizacije bi morali zabeležiti negativen trend zabeleženih hroščev, kar nakazuje vse večjo stabilnost aplikacije. [6]

3.4.3.2. Različica brez hroščev (RHB)

To je prva točka v projektu, kjer razvojni tim dohiti tim za testiranje. Tu ni več aktivnih hroščev ali pa noben ni več aktiven dlje časa (normalno 72 ur). Pričakovano je, da bo število hroščev po tej izdaji produkta naraslo, toda razvojni tim se bo trudil doseči RBH v naslednjih internih različicah. Prispetje do RHB je jasen znak, da je tim v zadnjih stopnjah projekta pred izdajo produkta. [6]

3.4.3.3. Kandidat za izdajo

Ko se tim odloči, da je produkt potencialno pripravljen za izdajo, se izdela kandidat za izdajo. Vsak kandidat za izdajo vsebuje vse elemente, ki so potrebni za izdajo produkta in nima aktivnih hroščev. Tim izvede intenzivno testiranje kandidata in tako odkrije še zadnje možne hrošče. Intenzivno testiranje ponudi odgovor, ali bo kandidat za izdajo postal končni produkt

ali pa mora tim izdelati novo različico kandidata za izdajo z ustreznimi popravki. Malo verjetno je, da bi bil prvi kandidat za izdajo že tudi končni produkt. [6]

3.4.3.4. Izdaja končnega produkta

Izdaja končnega produkta je različica kandidata za izdajo, za katerega se naročnik, stranka in celoten tim strinjajo, da je to različica, ki bo predstavljala končni produkt. To je različica, ki pomeni različico za na polico, za katero se dodaten razvoj ali testiranje ne izvaja več.
Določitev, katera različica predstavlja končni produkt, je težka odločitev. Končno je odgovor na to izdaja pravega produkta ob pravem času. Ali produkt v celoti izpolnjuje zahteve uporabnika? Ostali ključni elementi so poročilo o hroščih, rezultati testiranja kandidata za izdajo in stanje podpore produktu. [7]

3.4.4. Mejnik izdaja različice ter izdelki faze stabilizacije

Dosega mejnika *izdaja različice* je končni cilj projektnega tima. Predstavlja točko, ko je tim zaključil z delom na produktu in so tako produkt, njegovi elementi in drugi spremni izdelki pripravljeni za izdajo. To je tudi točka, kjer projektni tim prepusti lastništvo, dostavo in vzdrževanje nad aplikacijo sistemski skupini in skupini za podporo. S tem je tudi zaznamovano, da je projektni tim dosegel zastavljeno vizijo projekta.
Izdelki tega mejnika nudijo pomembne informacije za dostavo in uporabo produkta v njegovem produkcijskem okolju.
Za dosego mejnika so potrebni naslednji izdelki:

- **izdaja končnega produkta**
  Izvorna koda in izvršne datoteke.
- **opombe k izdanem produktu**
  Dokument, ki vsebuje informacije o izidu in zadnjih spremembah.
- **dokumenti podpore**
  Končne različice dokumentov podpore.
- **rezultati testiranja**
  Stanje hroščev in njihova baza za kasnejše projekte.
- **projektni arhiv**
  Vse pomembne informacije o produktu, ki so bile izdelane v fazi razvoja.
- **projektna dokumentacija**
  Projektna dokumentacija iz vseh faz projekta vključno z večjimi različicami ob večjih mejnikih. [1]

3.4.4.1. Opombe k izdanemu produktu

Vsak produkt vsebuje spremembe narejene v zadnjem hipu oz. druge elemente s katerimi morajo biti stranka, uporabniki in njihova podpora seznanjeni. S pripravo enostavnega dokumenta z omenjeno vsebino, se izognemo problemom, ki lahko nastanejo ob dostavi produkta s strani tima za dostavo. [1]

3.4.4.2. Uporabniški dokumenti

Ti so lahko različnih oblik od datotek pomoči, čarovnikov, vodičev za uporabo pa do priročnikov za tečaj. Osnovni namen teh dokumentov je dvojni, priprava uporabnikov na izdajo produkta in pomoč uporabnikov po njej. [1]

3.4.4.3. Projektni dokumenti

Ključ za dobro sprejetje izdanega produkta je ustrezna dokumentacija, namenjena uporabnikom. Projektni tim lahko vnaprej objavi skorajšnji zaključek projekta in

zainteresiranim uporabnikom ponudi lokacijo, kjer lahko najdejo nekatere predhodne dokumente.
Dobra ideja pred pripravo kompletne dokumentacije je predhodno priprava dokumenta, ki poudarja glavne lastnosti aplikacije. Ko se projekt nahaja na četrtini procesa dostave lahko tim na spletni strani objavi seznam pogostih vprašanj z odgovori. Seznam naj bo redno osvežen in v njegovo pripravo je potrebno vključiti uporabnike s svojimi dejanskimi primeri. [1]

3.4.4.4. Tečaj in priročniki za tečaj

Kot del procesa stabilizacije lahko projektni tim izdela tudi plan tečaja skupaj z ustreznim priročnikom. Tečaj je lahko izdelan v obliki samostojnega učenja, formalnega tečaja z več uporabniki ali pa neformalnega tečaja z eno osebo, odvisno od velikosti dostave in zapletenosti aplikacije. [1]

3.4.4.5. Dokumenti podpore

Sistemska dokumentacija je lahko sestavljena iz več delov, odvisno od aplikacije. Dokumentacija strežnika je potrebna, če je aplikacija sestavljena iz SUBP in podatkov. Ti dokumenti so lahko dodatno deljeni v dokumentacijo o spremembah in dopolnitvah k namestitvi, sistemskih nastavitvah in sprotnih nastavitvah (primer bi bil dokument, ki opisuje obnovo podatkov po podatkovni nesreči).
Odvisno od aplikacije, predvsem če je ta izdelana večnivojsko, je potrebno dokumentirati tudi možnosti nastavitev, parametrov in metodologij, ki naj bodo predhodno testirane v internih različicah.
Prav tako je potrebno izdelati dokumentacijo odjemalcev, ki vsebuje specifikacijo namestitve aplikacije in opravljene privzete nastavitve. Dokument naj bo pripravljen že v *razvojni fazi* in zaključen že v *fazi stabilizacije*. Poleg tega je za več nivojske aplikacije smiselno izdelati okvirno skico komunikacije med nivoji. [1]

3.4.4.6. Rezultati testiranja

Zbrati in trajno arhivirati je potrebno vse rezultate testiranja aplikacije. Kasneje lahko te dokumente uporabimo za določitev novih značilnosti produkta za novo različico. [1]

3.5. Dostava produkta

Ko je produkt končan in brez hroščev, ga predamo stranki. [2]

3.6. Programska orodja, ki pomagajo pri razvoju aplikacij

Za celoten proces nastanka aplikacije potrebujemo raznovrstna programska orodja. Dandanes si je brez njih nemogoče predstavljati celotni proces razvoja. Z njimi si olajšamo delo.

Programska orodja:

- Microsoft Visio,
- Microsoft Expression Blend,
- Microsoft Project,
- Microsoft Visual Studio.

3.6.1. Microsoft Visio

Microsoft Visio je programsko orodje, ki je namenjeno ustvarjanju diagramov (UML), gradnji podatkovnih modelov in je v pomoč pri risanju grafov. Z dinamični vizualnimi učinki, ki temeljijo na podatkih, si poenostavimo zapletenost. [8]
Podobni programi:

- Kivio ,
- IBM Rational Rose,
- Dia,
- ArgoUML ,
- StarUML .[9]

3.6.2. Microsoft Expression Blend

Expression Blend je profesionalno oblikovalsko orodje za izdelavo privlačnih namiznih in s spletom povezanih uporabniških izkušenj za okolje Windows. Z njim lahko izdelujemo prototipe aplikacij kot tudi končne aplikacije. [10]

Podobni programi:

- Visual Studio 2010,
- Sothink Quicker for Silverlight.

3.6.3. Microsoft Project

Informacijska podpora s programom Microsoft Project vodji projektov omogoča nazoren pregled nad projekti. Z diagrami postanejo zasedenost delavcev in njihove naloge v trenutku jasne. Bistveno se skrajša čas načrtovanja, izvajanja in kontrole projektov. Za učinkovit nadzor nad potekom dogodkov je dovolj le pogled, tudi v primeru dodatnih sprememb. Z večanjem obsega in števila projektov sicer narašča možnost za neporazdeljene obremenitve virov, vendar Project omogoča njihov optimalni izkoristek ali skrajšanje časa izvedbe projekta. Na predvidene aktivnosti nas redno opozarjajo opomniki. [11]

Podobni programi:

- OpenProj,
- GanttProject,
- Open Workbench. [12]

3.6.4. Microsoft Visual Studio

Microsoftov razvojni sistem Visual Studio je zbirka razvojnih orodij, ki razvijalcem programske opreme, pa naj gre za začetnike ali izkušene strokovnjake, pomaga premagovati kompleksne izzive in ustvarjati inovativne rešitve. Vloga zbirke Visual Studio je izboljšati razvojni proces ter omogočiti preprostejše uresničevanje inovacij. [13]

Podobni programi:

- Qt Creator,
- Code::Blocks,
- SharpDevelop,
- MonoDevelop. [14]

4. Expression Blend 3 – enostavna rešitev za izdelavo prototipov

Microsoft Expression Blend 3 je orodje, namenjeno oblikovalcem za gradnjo privlačnih (namiznih WPF ali spletnih Silverlight) aplikacij. Orodje je dokaj novo, zato je na začetku potrebna dodatna krivulja učenja, da lahko začnemo graditi smiselne in bogate aplikacije. Veliko razvijalcev in oblikovalcev se tako zateče k svojemu priljubljenemu ponudniku iskanja za raziskovanje primerov in navodil, kako se lotiti razvijanja ali oblikovanja v tem orodju.
Microsoft Expression Blend 3 (kot tudi veliko drugih Microsoft produktov) vsebuje veliko primerov uporabe ob zagonu orodja. Na začetnem oknu (Startup Window) ponuja precej privlačnih primerov uporabe, vodičev, ki se jih da izkoristiti. [15]
WPF in Silverlight zagotavljata platformo, ki prinese bogato, animirano, s predlogami aplikacijo. Zagotavljata zapleteno vektorsko grafiko in briše razliko med uporabniškim vmesnikom in vsebino. Ustvarimo lahko aplikacije s standardnim izgledom in si s tem poenostavimo razvoj. Na ta način dobimo več prožnosti in enostavnosti. Animacija in povezava s podatki sta možni skoraj povsod, tako da je aplikacija interaktivna. Lahko se uvozijo slike, video, 3D. Expression Blend je tudi v celoti napisan v WPF-ju. WPF je SDK, razvojna programska oprema, ki ni prijazna uporabniku. Tukaj pa pride v poštev Expression Blend. Blend je vizualno orodje za WPF in Silverlight in je oblikovalsko prijazno orodje, v katerem lahko uporabljamo celoten WPF. Omogoča risanje, animiranje, dostop do komponent in kontrol, povezovanje z bazo podatkov. Ima še veliko drugih funkcij, ki pripomorejo k boljši uporabniški izkušnji. Rišemo lahko tudi s pomočjo tablic za risanje. Uvažamo lahko slike, ki so v Photoshop formatu.
Kompleksne aplikacije imajo ponavadi veliko kode in funkcionalnosti. Expression Blend nudi popolno podporo popravljanja kode XAML, C# in vb. Na voljo je tudi »intellisense«, s katerim lažje programiramo. Projekt, ki ga ustvarimo v Expression Blendu, je možno odpreti tudi z Microsoft Visual Studiem. Možna je pretvorba prototip aplikacije v neko osnovo za nadaljevanje projekta v okolju Microsoft Visual Studio.

Slika 8: Expression Blend.

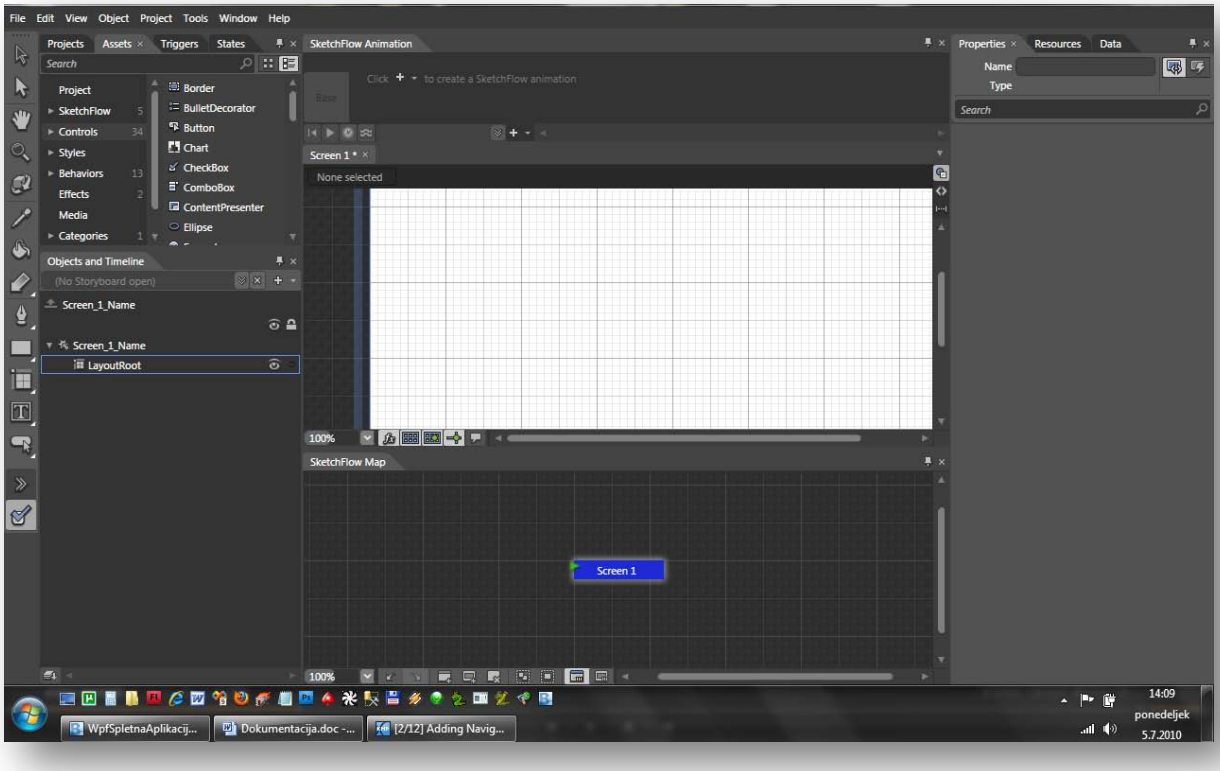
4.1. Expression Blend 4 in njegove novosti

Novosti v Expression Blend 4:

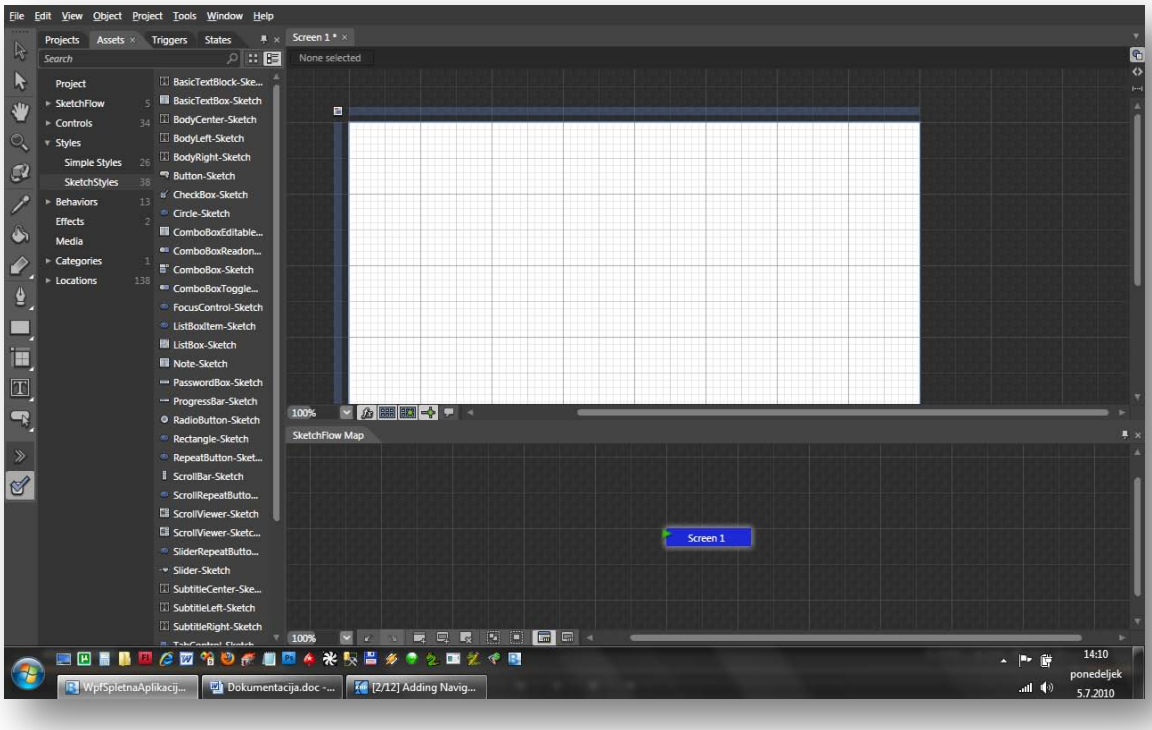
- Kompatibilnost s Silverlight 3 in WPF 3.5 z servisnim paketom (SP1),
- možnost objavljanja v Microsoft SharePoint,
- pretvorba povratnih informacij (feedback) v TFS delovno nalogo,
- menjavanje SketchStyle,
- možnost pavziranja in nadaljevanje SketchFlow animacij,
- izboljšano uvažanje Photoshop slik,
- nove kontrole,
- izboljšano oblikovanje kontrol,
- manj XAML kode. [16]

4.2. Prvi stik z novim okoljem in ureditev okolja na delo

Kreiramo nov projekt tako, da izberemo File→ New Project → »Wpf SketchFlow Application« oz. »Silverlight SketchFlow Application«. Vnesemo želeno ime in pritisnemo »OK«. Odpre se nam prazno okno. Prva stvar, ki jo naredimo, je ta, da spremenimo izgled delovne površine. To naredimo tako, da gremo na Window → Workspaces → Design in nato Windows → »Reset Current Workspace«, tako da dobimo osnovni delovni prostor. Nato si odstranimo tiste zavihke, ki jih ne potrebujemo. V našem primeru odstranimo »Object and timeline«, »SketchFlow Animation«, »Properties«, »Resources«, »Data«. Namen tega je, da si pustimo odprte samo tiste zavihke, ki jih trenutno potrebujemo. S tem si ustvarimo večjo delovno površino. S tem, ko zapremo določene zavihke, ni nič narobe, saj jih lahko vedno naknadno prikažemo. To storimo tako, da na meniju kliknemo »Windows« in tam imamo na voljo za obkljukati zavihke, ki jih potrebujemo. Ko si vse priredimo po naših željah, lahko delovno površino shranimo, tako da gremo na Window → »Save as New Workspace«, ga poimenujemo in shranimo. Tako ga lahko uporabimo tudi pri drugih projektih in ne zgubljamo časa s ponovnim nastavljanjem. Obstaja pa tudi bližnjica, ki navidezno zapre vse zavihke in tako nam ostane samo delovna površina. Ta bližnjica je tipka »F4«, ki nam prikaže (maksimira) oziroma skrije (minimira) zavihke. [17]



Slika 9: Delovno okolje pred popravki.

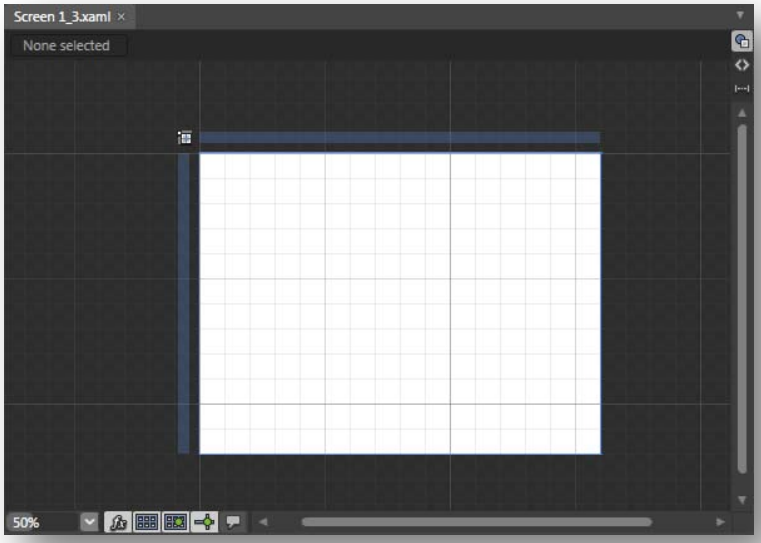


Slika 10: Delovno okolje po popravkih.

4.2.1. Predstavitev zavihkov

Obdelovalno okno

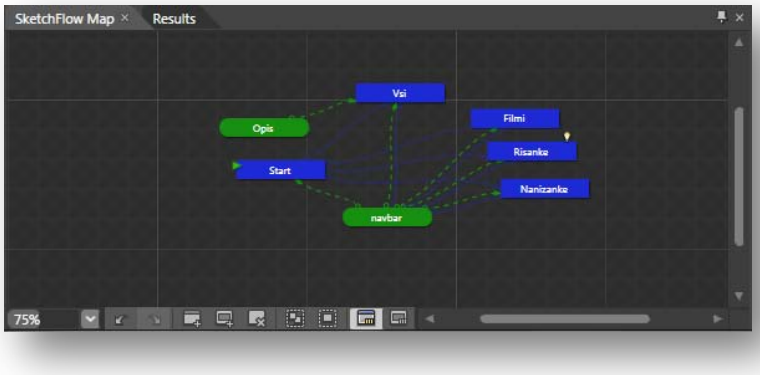
Kot pove že samo ime, tukaj prikazujemo želene podatke.



Slika 11: Zavihek Obdelovalnega okna.

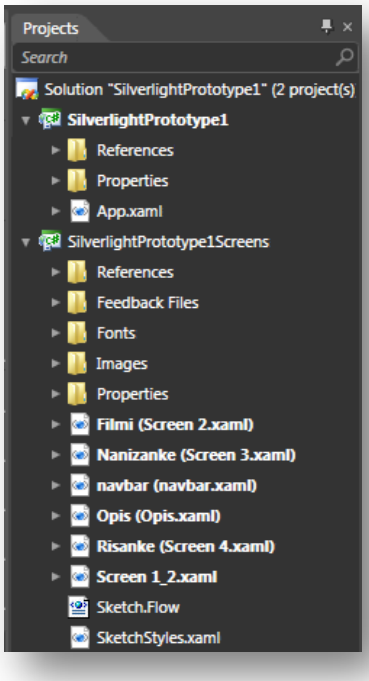
SketchFlow map

V SketchFlow map vidimo naš projekt v obliki nekakšnega miselnega vzorca. Imamo okna in komponentna okna, ki so povezana med sabo. Te povezave nam kažejo, kaj je med seboj povezano. S klikom na katerokoli okno oziroma komponentno okno se nam le to odpre.



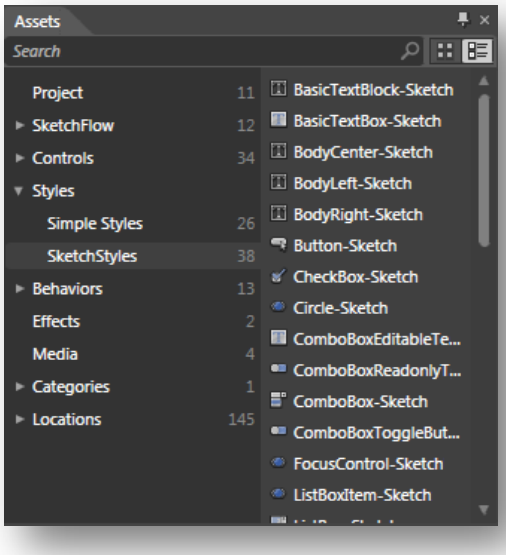
Slika 12: Zavihek SketchFlow Map.

Projects
Tukaj lahko neposredno dostopamo do vseh datotek, ki jih imamo v projektu.



Slika 13: Zavihek Projects.

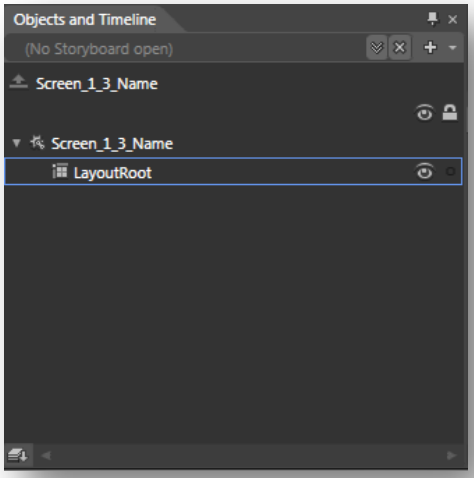
Assets
V Assets imamo na voljo veliko različnih kontrol za delo v Expression Blendu.



Slika 14: Zavihek Assets.

Object and Timeline

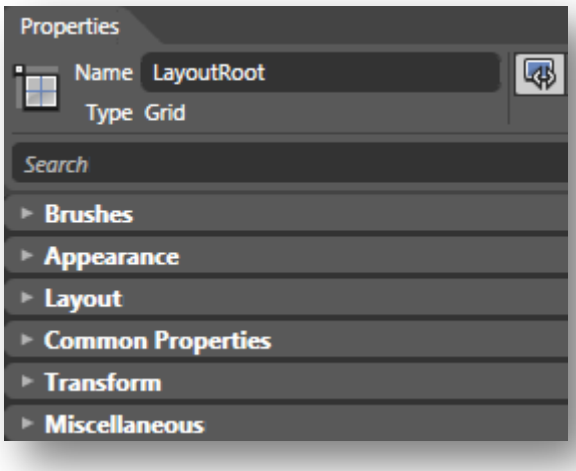
Tukaj imamo seznam kontrol, ki jih imamo trenutno na obdelovalnem oknu, in jim lahko nastavljamo različne lastnosti.



Slika 15: Zavihek Objects and Timeline.

Properties

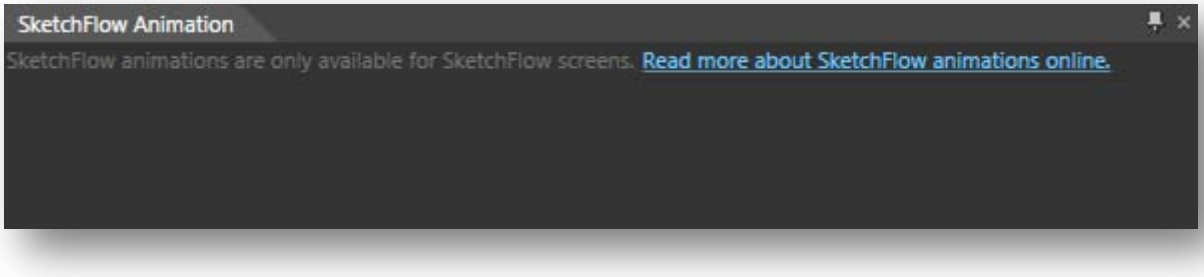
V Properies nastavljamo kontrolam razne lastnosti, akcije, stanje.



Slika 16: Zavihek Properties.

SketchFlow Animation

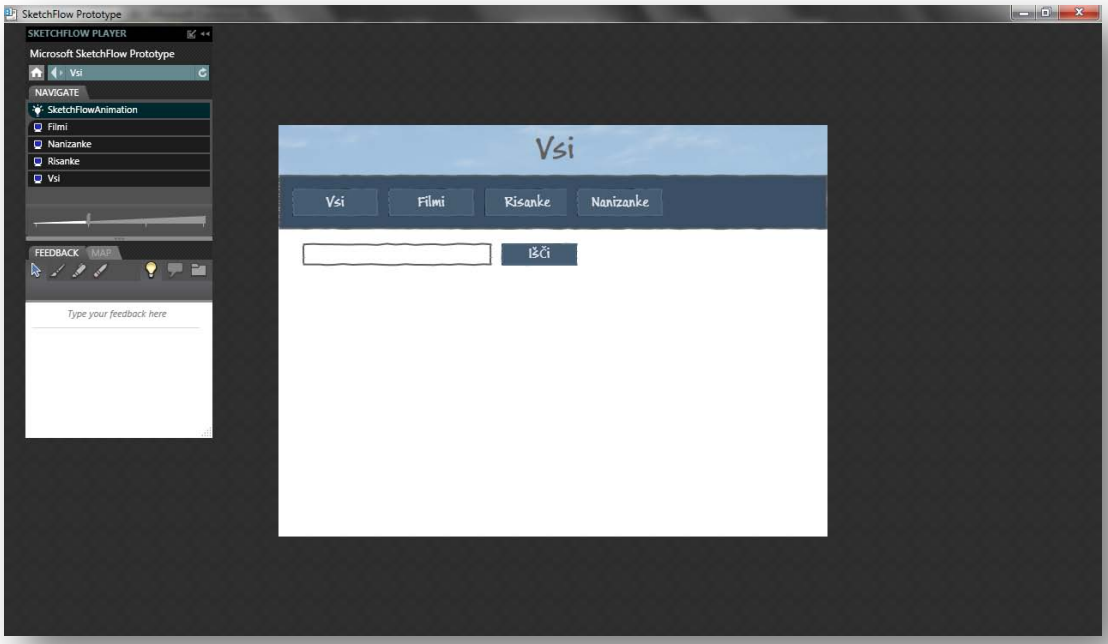
SketchFlow Animation je namenjen za izdelavo enostavnih animacij.



Slika 17: Zavihek SketchFlow Animation.

V Expression Blendu imamo še veliko pomembnih stvari, ki jih bomo spoznali tekom tega poglavja. Spoznali bomo še veliko drugih stvari kot so naprimer Feedback, Data, ki jih bomo prikazali tekom primera izdelave preproste spletne aplikacije.

4.3. Predstavitev SketchFlow player-ja

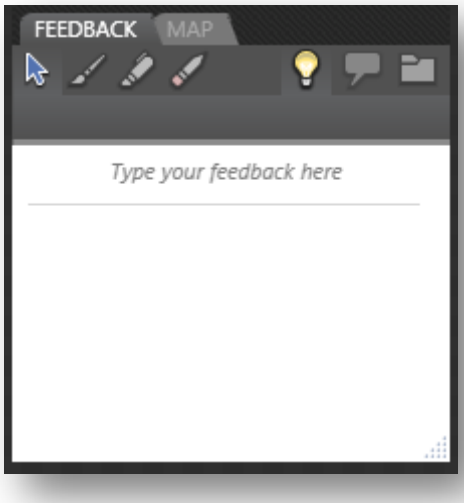


Slika 18: Predvajalnik SketchFlow player.

Ko zaženemo projekt (tipka F5), se nam odpre SketchFlow player. Znotraj player-ja lahko uporabnik navigira skozi različna okna. Z uporabo navigacijskega okna lahko uporabnik uporabi SketchFlow map za premikanje po različnih oknih.

Slika 19: Navigacijsko okno v predvajalniku SketchFlow player.

SketchFlow-player omogoča, da uporabnik daje razne povratne informacije s pomočjo ikone svinčnika ali označevalnika. To lahko shrani in ustvari se datoteka s končnico ».feedback«. Datoteko lahko pošlje želeni osebi in na ta način je komunikacija hitra. Kot primer lahko vzamemo izvajalca in stranko.

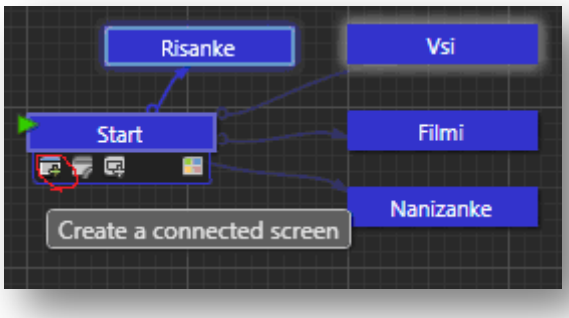


Slika 20: Feedback v SketchFlow playerju.

4.4. Izdelava prototipa preproste spletne aplikacije

4.4.1. Kratek opis aplikacije

Za izdelavo preprostega prototipa spletne aplikacije si bomo izbrali aplikacijo, kjer bomo imeli bazo filmov, nanizank in risank. Na voljo bomo imeli iskanje po vseh ali pa po posameznem atributu. Po izbiri določenega filma se nam bo prikazal poster tega filma, opis in možnost ogleda trailerja. Za izdelavo tega prototipa si bom pomagal z orodjem Expression Blend 3 in pomočjo Silverlight platforme.



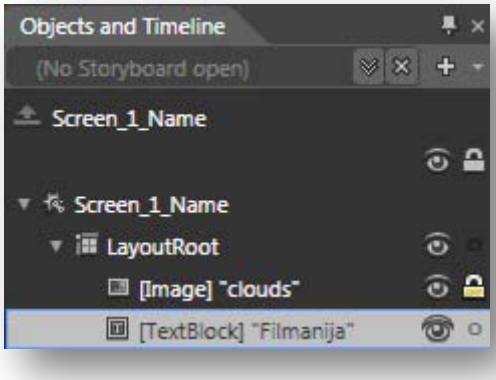
Slika 21: SketchFlow Map in nodi.

4.4.2. SketchFlow map in nodi

Najprej se lotimo dodajanje nodov (strani, oken). Posamezen node predstavlja svojo stran. Zato dodamo node (strani), ki jih bomo potrebovali in jih ustrezno poimenujemo.

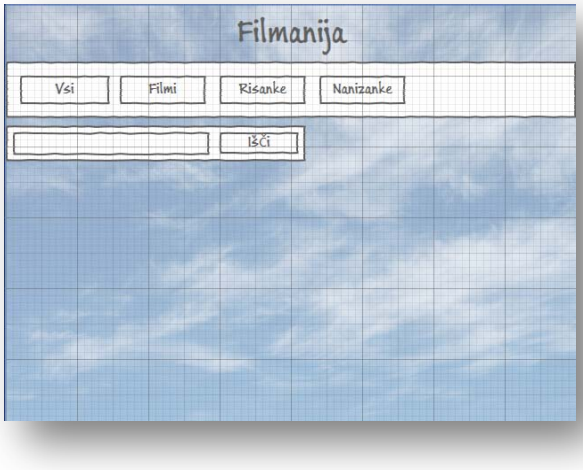
Kliknemo na vsak node in napišemo naslov. Na node »Start« želimo najprej dodati ozadje. Za to je potrebno klikniti na Project → »Add Existing item« in poiskati sliko, ki jo želimo uporabiti za ozadje. Če slika po uvozu še ni lepo poravnana, jo je potrebno poravnati tako, da

se prilega ozadju. Sliko bomo uporabili za ozadje, zato jo želimo nastaviti kot statično, da ne bi prišlo do neželenih premikov. Na voljo imamo možnost zaklepanja slike. To naredimo tako, da izberemo sliko in z desnim miškinim gumbom kliknemo in izberemo Order → »Send to Back«. Sliko lahko zaklenemo tudi na drugačen način. V zavihku »Objects and Timeline« vidimo vse, kar imamo trenutno na obdelovalnem oknu. Poiščemo sliko in kliknemo ikono kroga. To povzroči, da se slika zaklene (ikona ključavnice). Postopek ponovimo za vse strani, kjer želimo imeti takšno ozadje.



Slika 22: Kontrole, ki so trenutno na izbranem oknu.

Naslednja stvar, ki se je lotimo je dodajanje navigacijskih gumbov. Še preden se lotimo gumbov, pa prenesemo ozadje za gumbe. To najlažje storimo tako, da gremo na zavihek Assets → Syles → SketchStyle in izberemo »Rectangle-Sketch«. Na izbiro imamo možnost, da kliknemo in povlečemo kontrolo na okno. Lahko pa kontrolo izberemo in jo na oknu narišemo v poljubni velikosti (pravokotnik). Nato v ta pravokotnik povlečemo »Button-Sketch« (gumb). Za hitro kopiranje gumba samo držimo tipko »Alt«, kliknemo na gumb in ga premaknemo na želeno mesto. Tako ostane prvotni gumb na svojem mestu, kopirani gumb pa tam, kamor ga premaknemo. To ponovimo za vse gumbe. Na podoben način ustvarimo tudi polje za iskanje po bazi filmov. Dodamo »Rectangle-Sketch«, znotraj njega pa gumb Išči in »TextBlock-Sketch«.



Slika 23: Okno filmanija.

4.4.3. Komponentna okna

Če želimo imeti iskalnik in navigacijske gumbe na vseh straneh, jih lahko pretvorimo v kontrolo in to komponentno okno uporabimo na več straneh. Če so potrebni popravki, jih

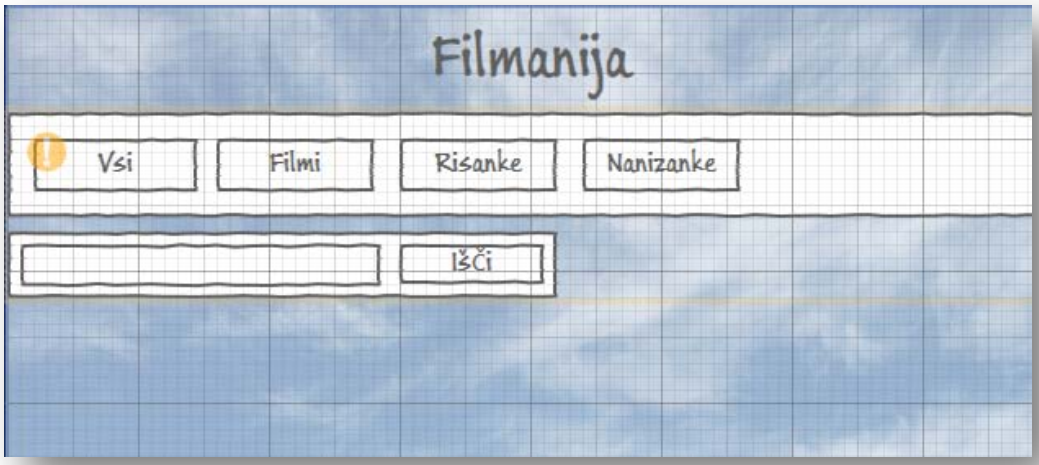
lahko naredimo samo na mestu, kjer se dejansko nahaja kontrola. Tako se izognemo napakam in skrajšamo čas popravljanja.

4.4.4. Kako do komponentnega okna

Najprej označimo vse elemente, ki jih želimo imeti v komponentnem oknu. V našem primeru označimo navigacijsko polje in iskalnik (glej zgornjo sliko). Z desnim klikom med možnostmi izberemo »Make Into Component Screen…«, poimenujemo in potrdimo datoteko. Ustvari se nam nov dokument (navbar.xaml). V »sketchflow map« vidimo novo ikono »navbar«, ki se razlikuje od ostalih. Običajna okna imajo obliko pravokotnika, komponentna okna pa so takšne oblike, kot je prikazano na spodnji sliki.

Slika 24: Ikona komponetnega okna navbar.

Če želimo uporabiti to komponetno okno »navbar«, ga izberemo in povlečemo na vsa okna, kjer ga potrebujemo. Tako naredimo črtkane povezave do oken. Komponentno okno prepoznamo na navadnih oknih po ikoni klicaja kot je razvidno iz spodnje slike.



Slika 25: Komponetno okno navbar na oknu filmanija.

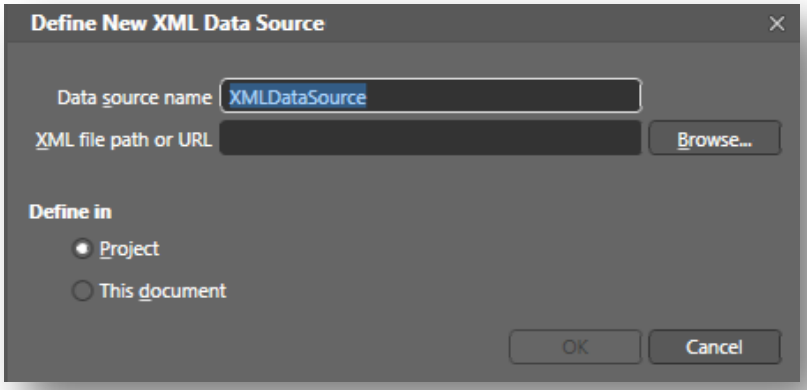
4.4.5. Gumbi in akcije

Če želimo, da imajo gumbi dejansko neko uporabnost nas mora posamezen gumb presmeriti na želeno okno. To naredimo tako, da odpremo komponentno okno »navbar«. Pred nami se odpre komponentno okno z navigacijskimi gumbi. Označimo gumb »Vsi« in kliknemo desni gumb na miški. Z miško gremo na »Navigate to« in izberemo »Vsi«. S tem, ko smo to izbrali smo dodali akcijo na ta gumb. Kadar bomo kliknili na ta gumb, se nam bo odprla stran »Vsi«. Isto naredimo za gumb (Filmi, Risanke, Nanizanke).

4.4.6. Iskalnik in xml podatki

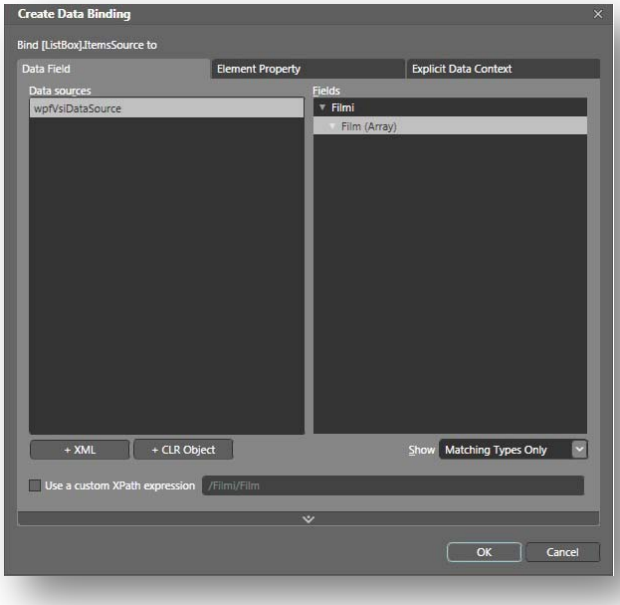
Za rezultat iskanja po filmih bomo potrebovali »ListBox-Sketch«. Ta se pa nahaja pod Assets → SketchStyles → »ListBox-Sketch«. Izberemo in povlečemo na želeno okno. Sedaj imamo

»ListBox«, ki ga želimo napolniti s želenimi podatki. To storimo tako, da označimo »ListBox« in kliknemo desni klik pri tem pa izberemo »Bind ItemSource to Data«. Kliknemo na »+xml«. Odpre se okno »Define New XML Data Source«. S klikom na »Browse…« poiščemo že vnaprej pripravljen XML napolnjen s podatki in ga izberemo.



Slika 26: Prikazno okno za izbiro xml podatkov.

Sedaj vidimo okno kot je prikazan na spodnji sliki. Kliknemo na zavihek »DataField« in pod »Data Sources« izberemo »wpfVsiDataSource«. V Polju »Field«s pa pošičemo po korenu navzdol »Film(Array)« in ga izberemo. Ko to naredimo potrdimo in naš »ListBox« bi moral prikazati podatke katere smo imeli v XML-ju.



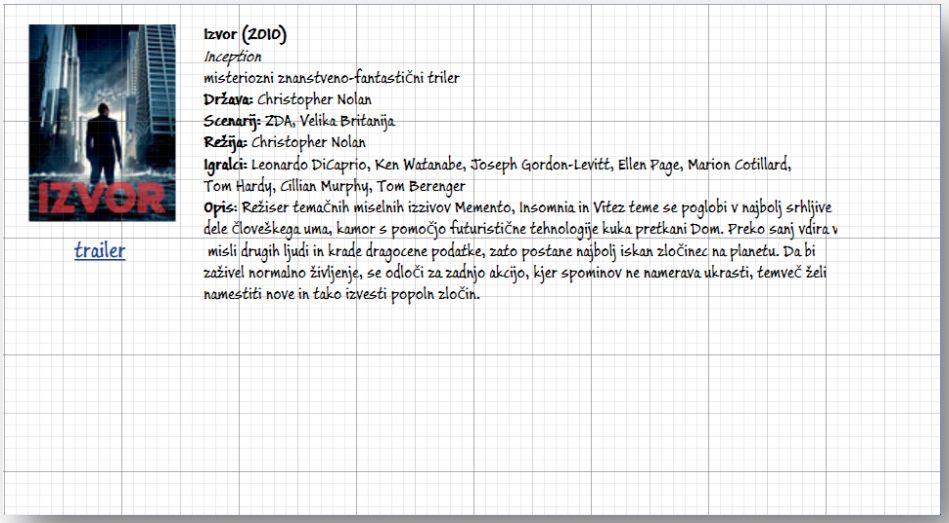
Slika 27: Prikazno okno za izbiro točno določenih podatkov.

Prikaže se nam »ListBox-Sketch«, napolnjen s podatki.

4.4.7. Okno Opis in primer opisa filma

V »SketchFlow map« dodamo novo komponentno okno. To naredimo tako, da kliknemo desni gumb na miški, izberemo »Create Component Screen« in ga poimenujemo kot »Opis«. Odpre se nam prazno okno, na katerem bomo prikazali opis filma. Gremo na »Object and Timeline« in preimenujemo okno, da dobi prijaznejše ime. Spremenimo ime

»Screen_1_5_Name« v Opis. Izberemo okno in v Properties → »Layout« nastavimo višino in širino okna na »640x350«. Sledi uvoz prej pripravljene slike (v našem primeru bomo naredili primer za film Izvor). Zatem v Assets → »SketchStyles« poiščemo »TextBlock-Sketch« in ga prenesemo na komponento. »Textblock« razširimo toliko, da je dovolj prostora za besedilo, ki ga bomo vnesli. Vnesemo želene podatke o filmu. Pod sliko vstavimo še »TextBlock-Sketch«, ki bo namenjen za prikaz trailerja. Ime spremenimo v trailer. V »Properties« mu spremenimo barvo v modro in mu damo podčrtan slog, da bo videti kot gumb.

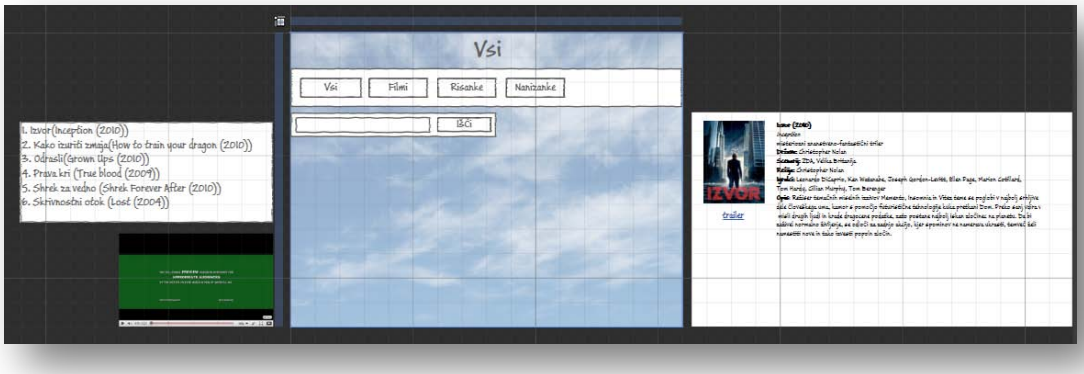


Slika 28: Okno opis, ki prikazuje opis filma.

4.4.8. Animacija iskanja in izbira filma

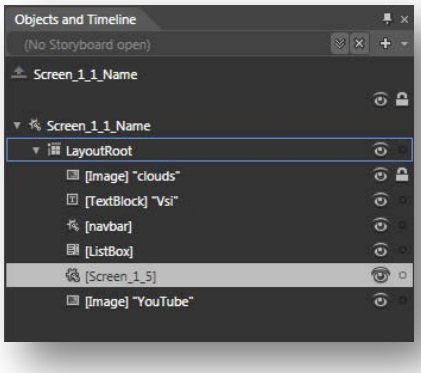
Najprej označimo »ListBox-Sketch« (rezultat iskanja) in v »Properties« pod »Apperance« nastavimo »Opacity« na 0%.

Nato povežemo komponetno okno »Opis« s stranjo »Vsi«, tako da se nam prikaže komponenta »Opis« na strani »Vsi«. Poravnamo jo tako, da nam ustreza. Tako kot pri »ListBox-Sketch« nastavimo »Opacity« na 0%. Pripnemo še sliko Youtube predvajalnika in jo premaknemo na želeno pozicijo. Nastavimo »Opacity« na 0%. Pripnemo še sliko miškinega kurzorja, vendar tukaj pustimo »Opacity« na 100%. Umaknemo sliko iz vidnega polja okna »Opis«. To naredimo zato, ker bomo glede na animacijo prikazovali in skrivali določene stvari in bomo za bolj realističen izgled premikali sliko kurzorja.



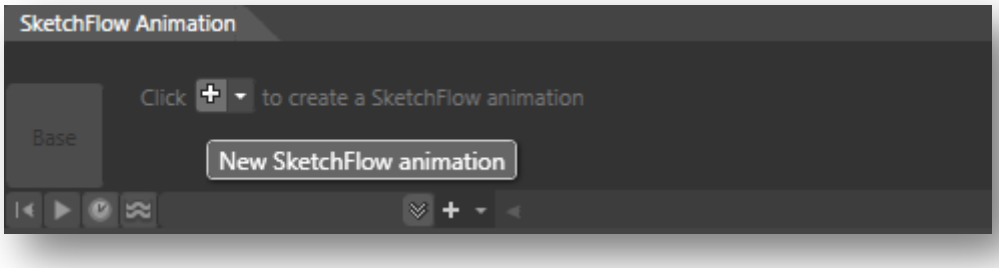
Slika 29: Kontrole, ki bojo uporabljene za animacijo.

Za kasnejše lažje izbiranje »ListBox-a« in ostalih predmetov si bomo pomagali tako, da jih bomo izbirali v »Objects and Timeline«.



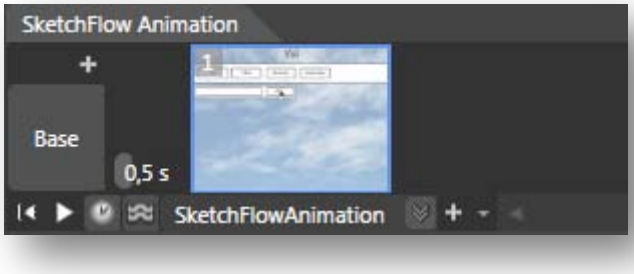
Slika 30: Kontrole, ki so trenutno na izbranem oknu.

Če še nimamo odprtega zavihka za »SketchFlow Animation«, gremo pod »Window« in izberemo »SketchFlow Animation«. Tako se nam prikaže zavihek, ki ga potrebujemo za animacijo. Kliknemo na plus.



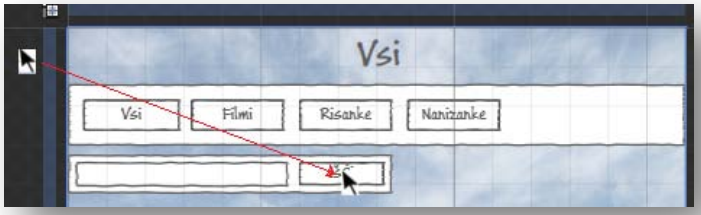
Slika 31: Prva stvar, ki jo naredimo pri animaciji.

S tem, ko kliknemo na plus, se nam odpre okence za animacijo, ki je po defaultu nastavljena na trajanje 0,5s. To lahko spremenimo po želji. Rdeča obroba in napis »SkectFlow animation Frame_1 recording is on« nas opozarjata, da imamo vključeno snemanje.



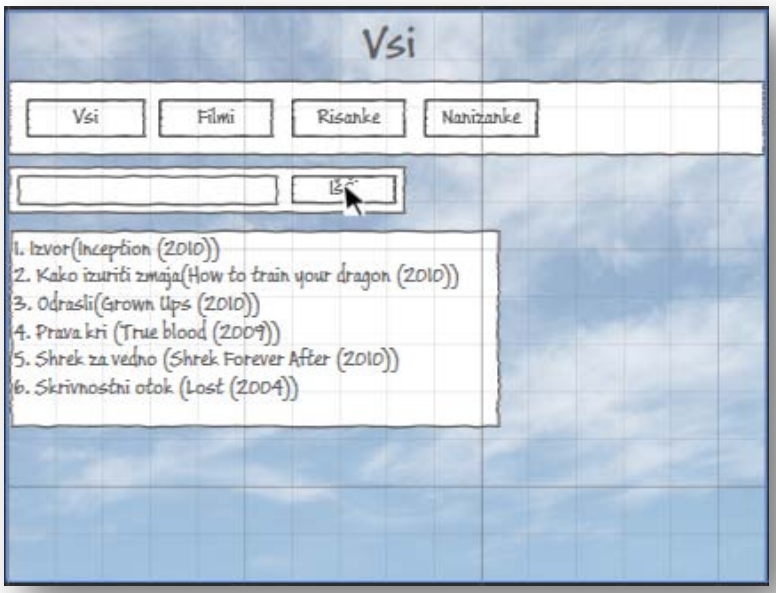
Slika 32: Prikaz slike po kliku na ikono plusa.

V naslednji potezi želimo premakniti sliko miškinega kurzorja na gumb Išči. S tem bomo ponazorili premik na gumb in klik. To je prvi del animacije.



Slika 33: Prikaz premika prvega dela animacije.

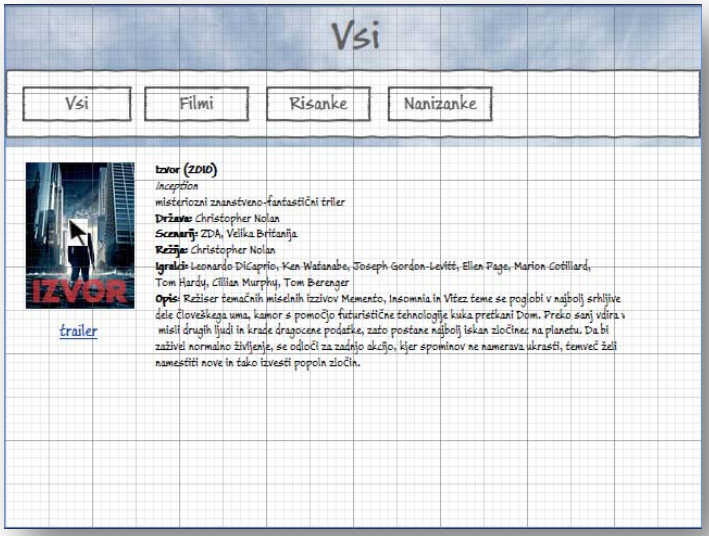
Drugi del animacije naredimo tako, da se postavimo v zavihek »SketchFlow animation«, kliknemo desni gumb na miški in izberemo »Insert New«. Sedaj lahko naredimo drugi del animacije. Gremo v zavihek »Object and Timeline« in izberemo »ListBox-Skecth« in mu nastavimo »Opacity« na 100%, da prikažemo »ListBox«.



Slika 34: Prikaz rezultata iskanja.

Tretji del animacije naredimo tako, da spet izberemo »Insert New«. Premaknemo sliko kurzorja na film Izvor in s tem smo zaključili tretji del animacije.

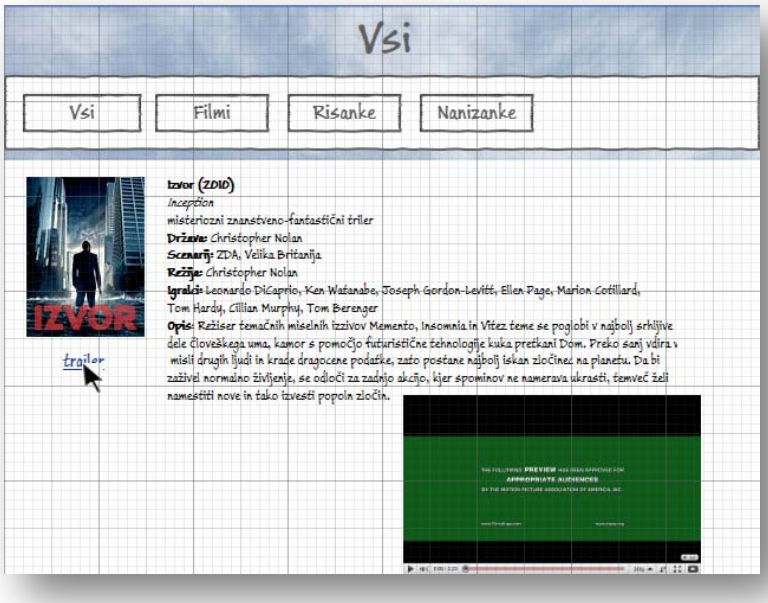
Pri četrtem delu animacije nastavimo »ListBox-Sketch« (rezultat iskanja) »Opacity« na 0% in komponentnemu oknu Opis nastavimo »Opacity« na 100%. Prikaže se nam opis filma Izvor.



Slika 35: Prikaz opisa filma.

Za peti del animacije ponovno izberemo »Insert New« in premaknemo sliko kurzorja na napis trailer.

Za šesti del animacije spet izberemo »Insert New« in sliki YouTube nastavimo »Opacity« na 100%. Prikaže se nam slika trailerja.



Slika 36: Prikaz trailerja.

4.4.9. Predloga

V Expression Blend-u imamo na voljo predloge, lahko pa vsako stvar oblikujemo sami. Lahko imamo naprimer več gumbov, ki imajo enak stil oblikovanja. Popravki so potrebni samo enkrat, da vsem gumbom spremenimo slog. Na voljo pa imamo tudi možnost, da vsak gumb spreminjamo posebej. Če želimo imeti isti slog za vse gumbe, moramo spreminjati nastavitve v točno določeni datoteki (SketchStyle.xaml). Poiščemo na primer »Button-

Sketch« in nastavimo želene nastavitve v »Properties«. V primeru, da želimo nek gumb, ki bo drugačen od ostalih, pa se lotimo spreminjanja tam, kjer smo ga prenesli na obdelovalno okno. Gremo v »Properties« in ga spremenimo po potrebi.

4.4.9.1. Spreminjanje predloge

Za spreminjanje predloge, moramo poiskati zavihek »Resources« in v »SketchStyle.xaml« imamo vse kontrole, ki jih lahko spreminjamo.

Button-Sketch

Poiščemo »Button-Sketch« in ga izberemo. V zavihku »Properties« mu najprej nastavimo širino in višino na 100 x 32. Nato nastavimo barvo gumba »Background« (#FF465C71), »BorderBrush« (#FF4E667D), »Foreground« (#FFDDE4EC).

Rectangle-Sketch

Poiščemo »Rectangle-Sketch« in ga izberemo. V zavihku »Properties« mu spremenimo »Background« (#FF524E75).

Navbar.xaml

Naknadno smo se odločili, da odstranimo »Recktangle-Sketch« pri iskalniku zaradi izgleda, zato ga odstranimo.

Ozadje

Ozadje, ki je trenutno ozadje cele strani, bi vidno samo pri naslovu, ostalo ozadje bo pa belo.

ListBox-Sketch

Za »ListBox« želimo nastaviti prosojno ozadje, na katerem se vidi besedilo. »BorderBrush« nastaviti na belo barvo.

4.4.10. Izvoz projekta in povratne informacije

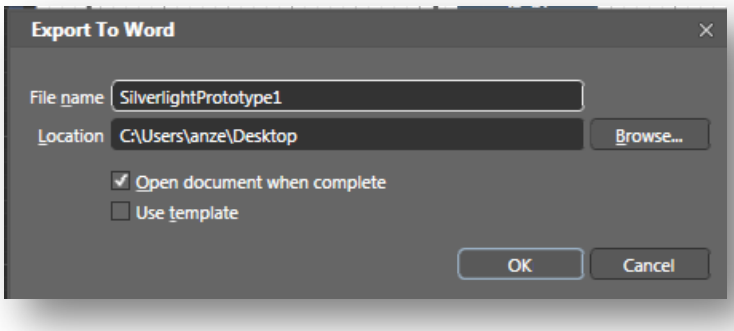
Ob zaključku projekta oziroma ob potrebi po tem, da stranko seznanimo s trenutnim stanjem projekta, lahko projekt enostavno izvozimo in pošljemo stranki.

4.4.10.1. Izvoz projekta

Za izvoz je potrebno izbrati Project → »Package SketchFlow Project«. Pokaže se nam okno, kjer imamo na voljo izbiro ime in pot, kamor naj se datoteke shranijo. Ko to potrdimo, se nam zgenerira mapa z našim projektom, ki jo stisnemo in pošljemo stranki.

4.4.10.2. Izvoz datoteke word

Na voljo imamo tudi izvoz projekta v Word datoteko. Izberemo File → »Export to Microsoft Word«. Prikaže se nam okno za ime datoteke in pot, kamor bomo datoteko shranili. Na voljo imamo, da se nam datoteka odpre takoj, ko jo shranimo in na voljo imamo tudi, da uporabimo vnaprej pripravljene predloge. Datoteka vsebuje kazalo vsebine, kazalo slik ter vse slike oken in komponent, ki jih imamo v projektu. To datoteko lahko tudi pošljemo stranki v pregled oziroma jo lahko uporabimo tudi kot nekakšen osnutek za dokumentacijo.

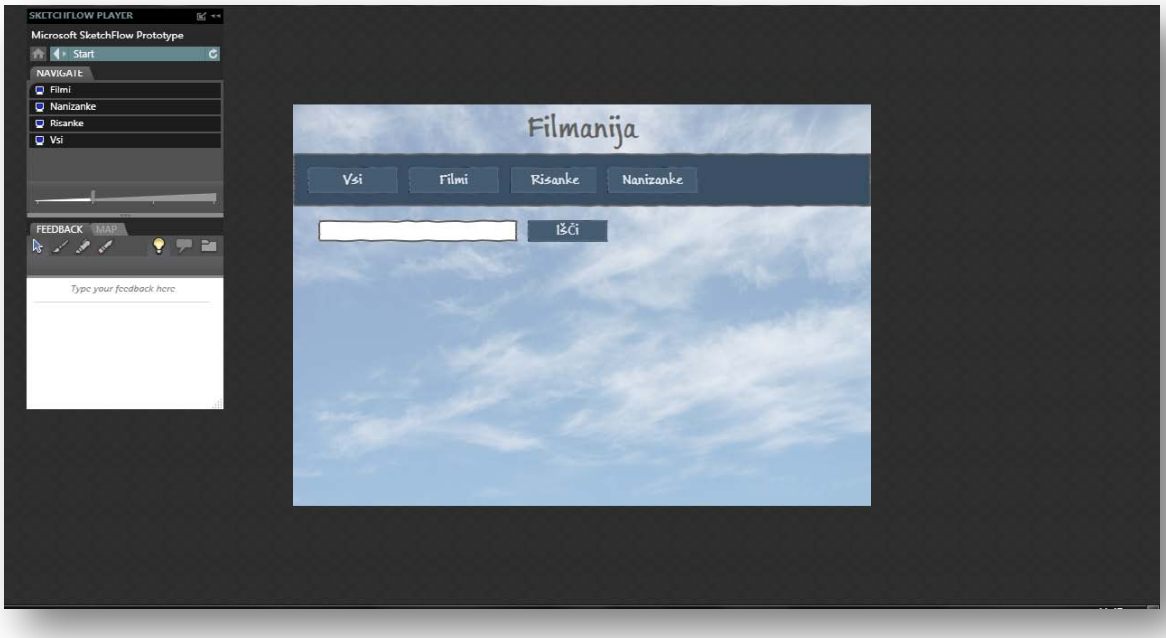


Slika 37: Prikazano okno za izvoz projekta v word dokument.

Imamo tudi, da uporabimo vnaprej pripravljen predloge. Datoteka vsebuje kazalo vsebine, kazalo slik, vse slike oken in komponent, ki jih imamo v projektu. To datoteko lahko tudi pošljemo stranki v pregled oz. lahko uporabimo tudi kot nekakšen osnutek za dokumentacijo.

4.4.11. Povratne informacije (Feedback)

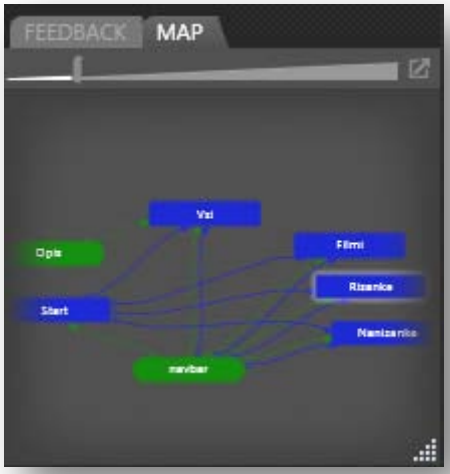
Ko stranka dobi stisnjeno datoteko projekta, jo razpakira in klikne na TestPage.html datoteko. Odpre se ji SketchFlow predvajalnik, kjer je naš projekt.



Slika 38: SketchFlow player.

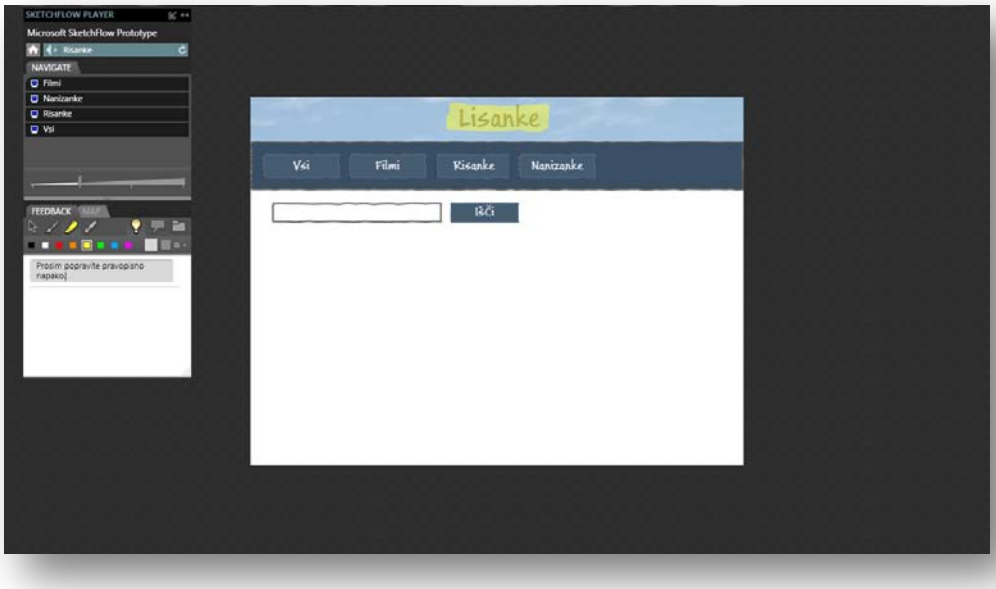
Stranka si ogleda projekt in ko opazi kakšno napako oziroma pomanjkljivost, lahko na enostaven način to tudi prikaže na SkecthFlow predvajalniku.

V našem primeru gre stranka na zavihek Map in opazi pravopisno napako, kjer namesto naslova »Risanke« piše »Lisanke«.



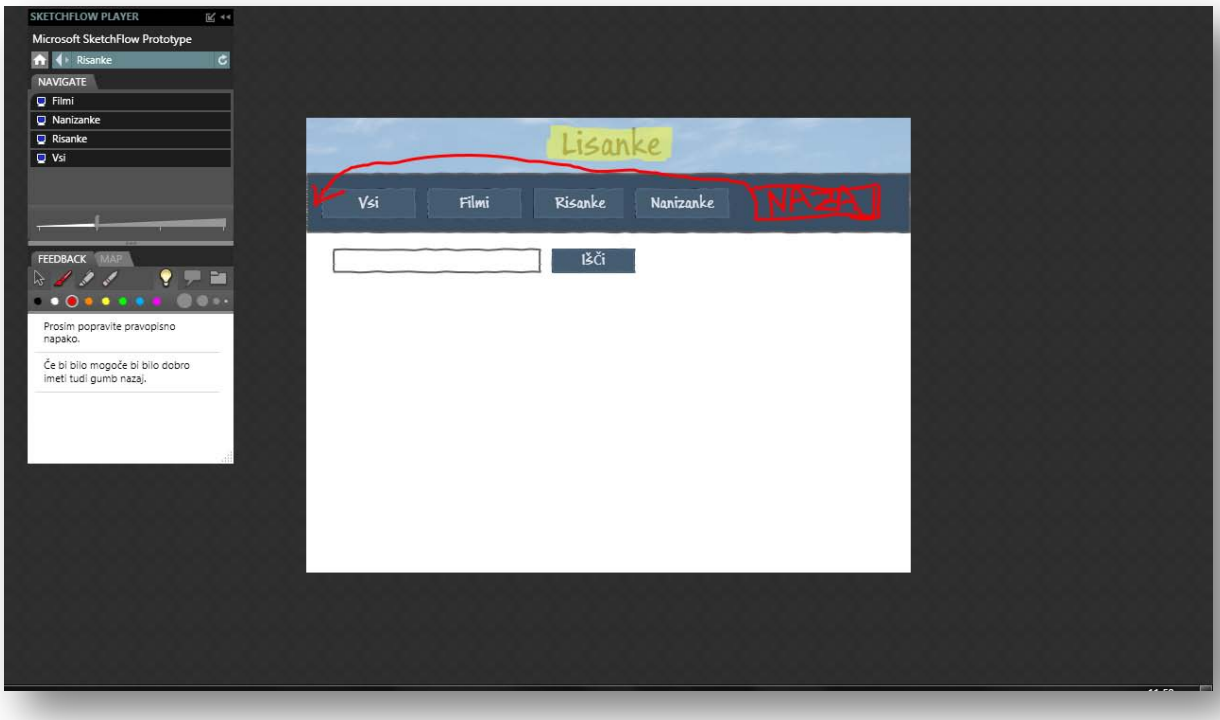
Slika 39: SkectFlow player map.

Stranka se pomakne na zavihek »Feedback«, kjer izbere ikono pisalo označevalnik in označi besedilo »Lisanke« in napiše opombo pod zavihkom »Feedback«, kot je prikazano na spodnji sliki.



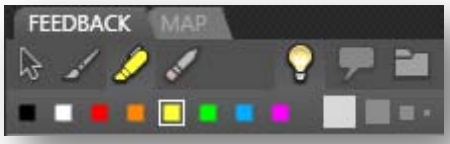
Slika 40: Prikaz napake 1.

Stranka je opazila še eno pomanjkljivost. Predlaga še možnost dodatnega gumba, s katerim bi se lahko pomikali nazaj. Izbere čopič za risanje in nariše gumb. S puščico prikaže, kje naj se ta gumb nahaja.



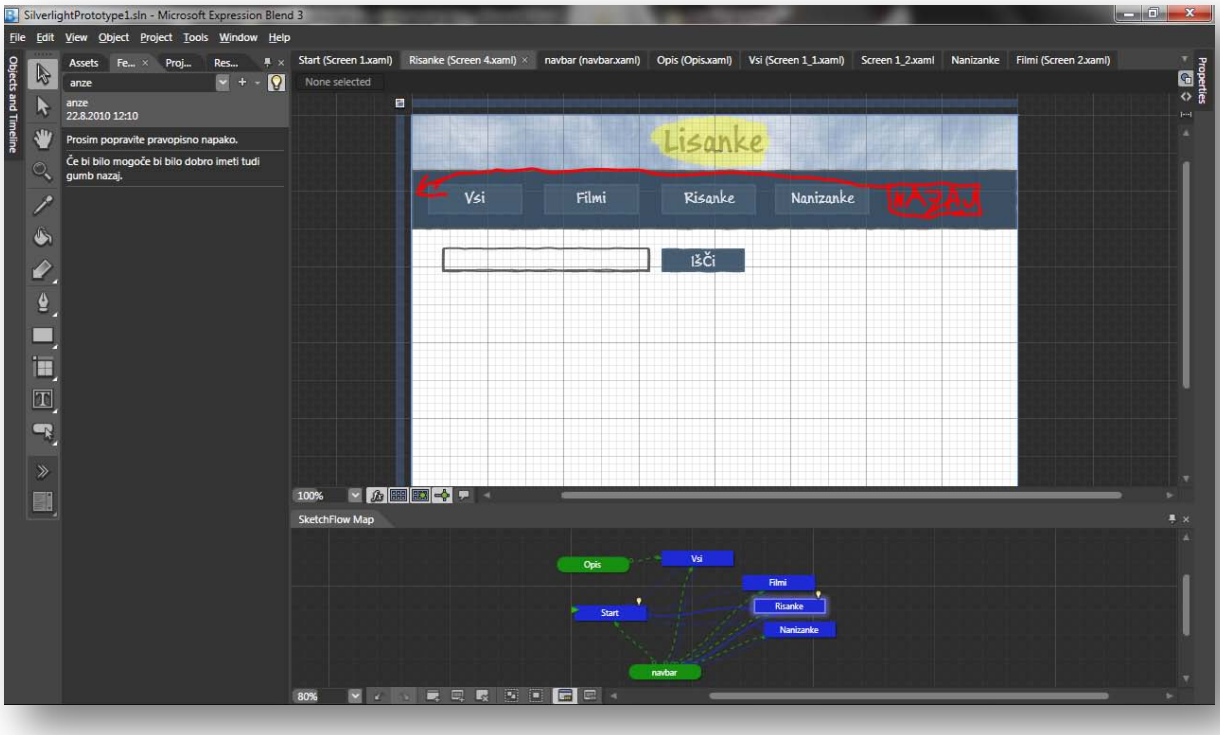
Slika 41: Prikaz napake 2.

V primeru, da stranka želi pobrisati, kar je naredila, ima na voljo tudi radirko za brisanje. Na voljo ima še navidezno skrivanje vsega, kar je dopisala. Klikne na svetilko in vsi popravki se skrijejo.



Slika 42: Feedback orodja.

Ko je stranka zadovoljna s popravki, lahko projekt enostavno izvozi. Klikne na ikono mape, ki je prikazana na zgornji sliki (desna ikona). Ponudi se ji izvoz feedback-a ali pa resetiranje. V našem primeru bo stranka izvažala. Ko izbere izvoz, se prikaže okno, kjer avtor napiše povratne informacije ter svoje ime oziroma kratico. Na podlagi tega bo programer oziroma vodja projekta vedel, koga lahko kontaktira za bolj podrobne informacije. Ko vnesemo ime avtorja, izberemo še ime datoteke in pot, kamor bomo datoteko shranili. Shrani se datoteka s končnico feedback. Ta datoteka je zelo majhna. Gre le za eno plast, ki jo lahko uvozimo v projekt in jo na enostaven prikažemo oziroma skrijemo. Stranka nam zdaj pošlje to datoteko. Ko jo dobimo, jo to uvozimo v projekt. To naredimo tako, da gremo na Window → »Feedback«. Odpre se nam zavihek »Feedback«. Kliknemo na plus znak in poiščemo datoteko s končnico feedback. V zavihku »Feedback« se nam prikažejo vse opombe, avtor in čas. Na oknih, kjer je stranka naredila popravke, se nam popravki prikažejo tudi vizualno. V »SketchFlow map« vidimo tudi, na katerih oknih so bile narejene spremembe oziroma opombe. To vidimo po ikoni v obliki žarnice. Gre le za dodatno plast, ki jo lahko poljubno skrijemo ali prikazujemo s klikom na ikono žarnice v zavihku »Feedback«.



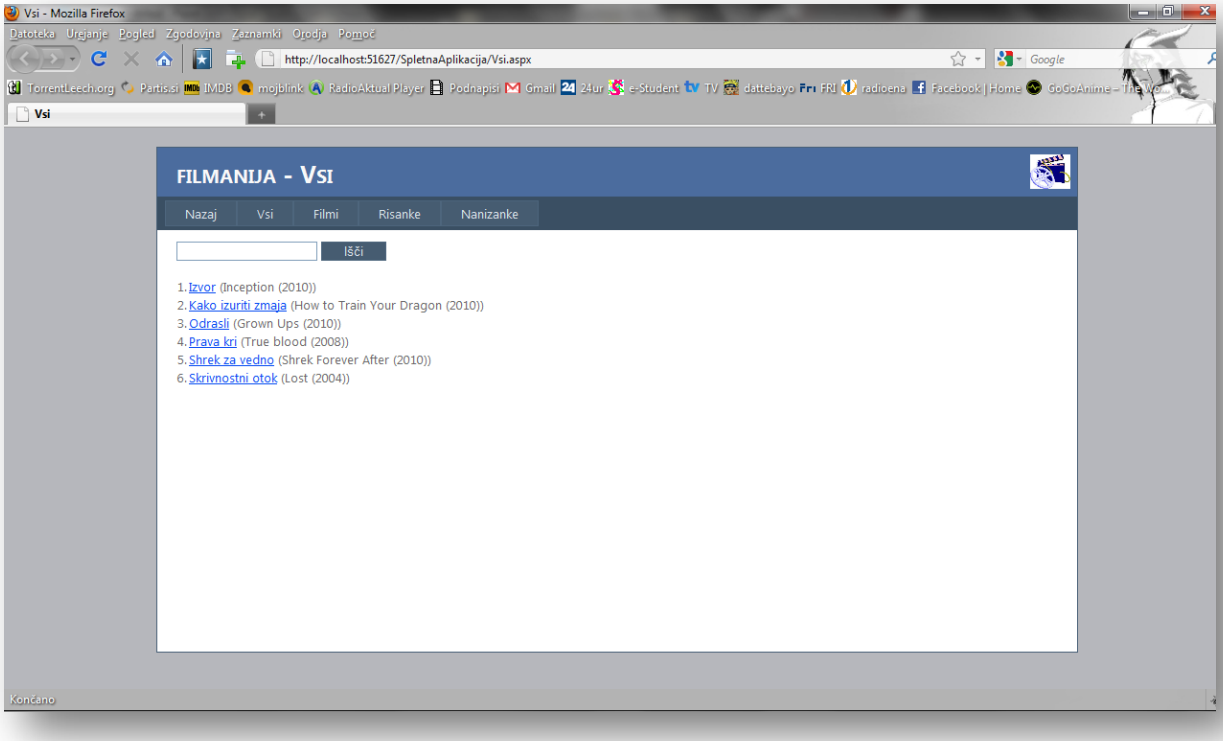
Slika 43: Uvoženi feedback.

Če zaženemo projekt, vidimo popravke tudi v predvajalniku. Dodamo lahko svoj »feedback« in ga po istem postopku, kot je opisano zgoraj, pošljemo stranki nazaj.

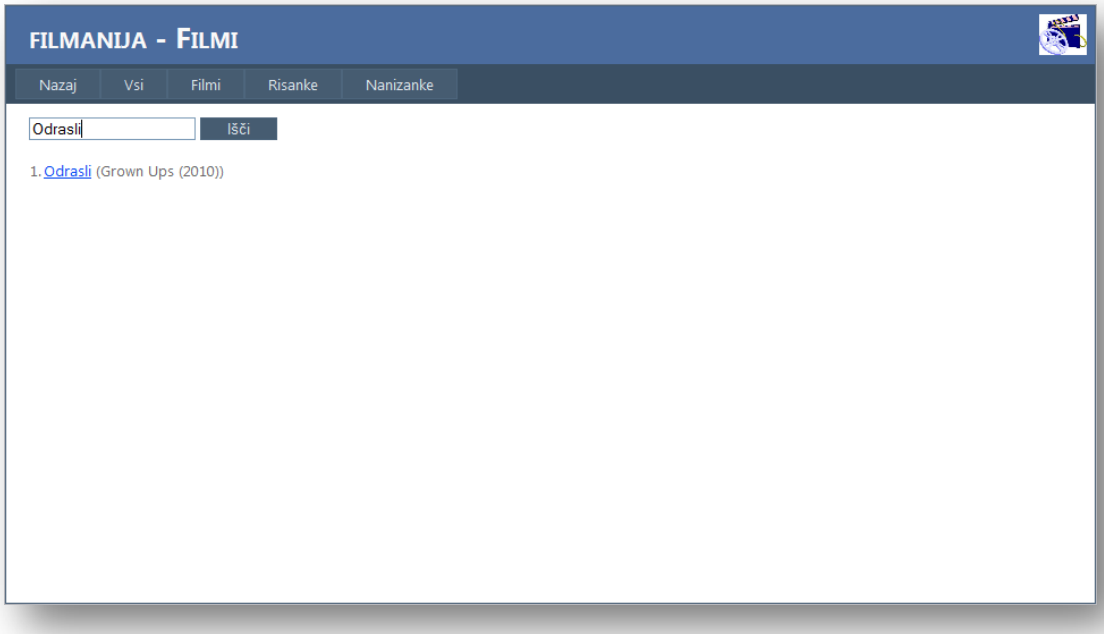
4.5. Preprosta spletna aplikacija

Ko je prototip dokončan, sledi izdelava dejanske aplikacije. Za izdelavo spletne aplikacije smo si izbrali orodje Microsoft Visual Studio 2010. Spletna aplikacija bo narejena s pomočjo ASP.NET in programskega jezika C#. Program bo vseboval tudi podatkovno bazo. Za podatkovno bazo smo izbrali Microsoft SQL Server 2008.

4.5.1. Slike končne aplikacije



Slika 44: Spletna aplikacija - osnovna stran (seznam filmov).



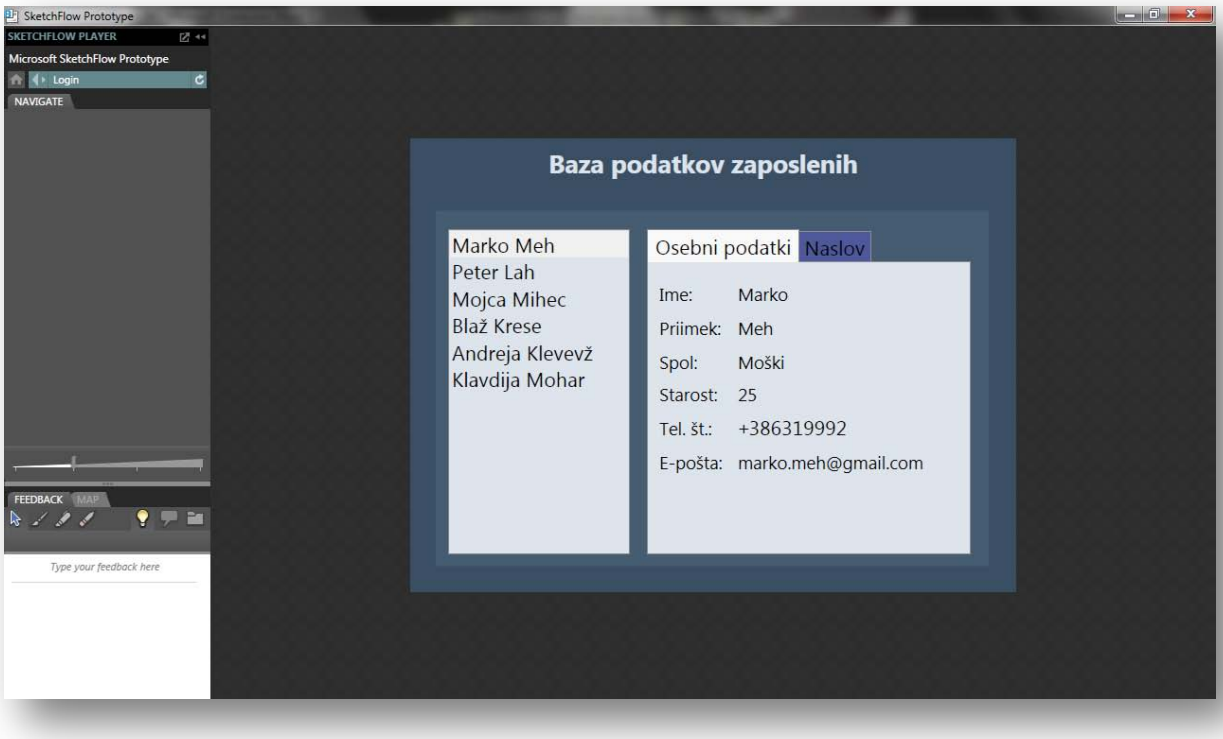
Slika 45: Iskanje filma v zavihku Filmi.

Slika 46: Opis filma Odrasli.

Slika 47: Prikaz trailerja filma Odrasli.

4.6. Preprosta WPF prototipna namizna aplikacija

Za izdelavo preprostega prototipa namizne aplikacije si bomo izbrali aplikacijo, kjer bomo imeli na voljo bazo podatkov o zaposlenih ter podrobne podatke. Za izdelavo tega prototipa si bomo pomagali z orodjem Expression Blend 3. Prototip bo temeljil na WPF platformi. Pri izdelavi aplikacije bomo čim bolj usmerjeni k temu, da bomo kasneje lahko ta prototip uporabili za kasnejše razvijanje aplikacije. Zato bomo uporabili kontrole, ki vizualno niso kot skice, temveč takšne kontrole, ki bodo že imele končni izgled. Izgled prototipa lahko vidimo na spodnji sliki.



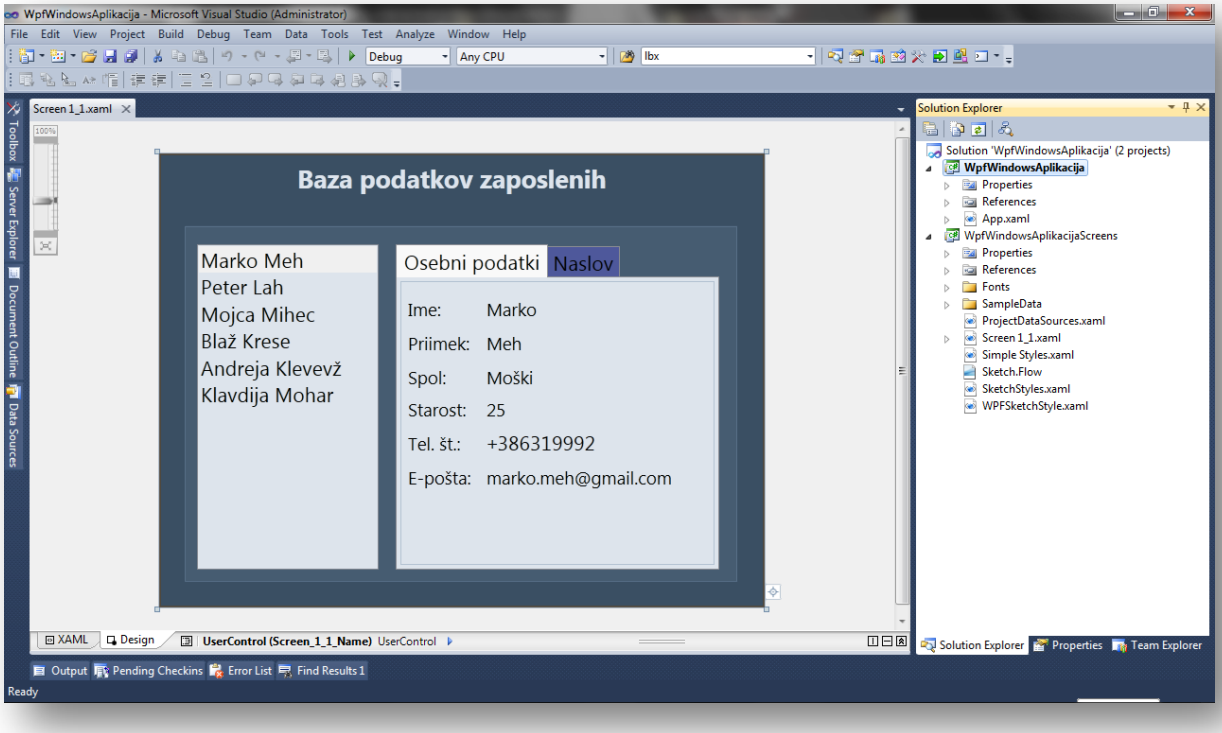
Slika 48: WPF prototip in aplikacija pred pretvorbo.

4.7. Pretvorba WPF prototipa v WPF aplikacijo.

Pretvorba WPF prototipa v WPF aplikacijo je namenjena temu, da ta prototip pretvorimo v WPF aplikacijo, ko je stranka zadovoljna z njegovim izgledom. Po pretvorbi dodamo še dodatne funkcionalnosti, ki se jih ni dalo realizirati v Expression Blendu in tako dobimo končno različico aplikacije. Niti Microsoft Expression Blend 3, niti Microsoft Expression Blend 4 trenutno ne omogočata enostavne pretvorbe prototipov. Pretvorba se pri obeh orodjih izvede na enak način. Prikaz pretvorbe WPF prototipa v WPF aplikacijo je prikazan v nadaljevanju.

4.7.1. Prikaz pretvorbe WPF prototipa v WPF aplikacijo

1. Še preden se lotimo pretvorbe, je pomembno, da naredimo varnostno kopijo aplikacije. Nato projekt prototipa, ki smo ga razvili s pomočjo Expression Blend-a, odpremo s pomočjo Microsoft Visual Studia 2010.



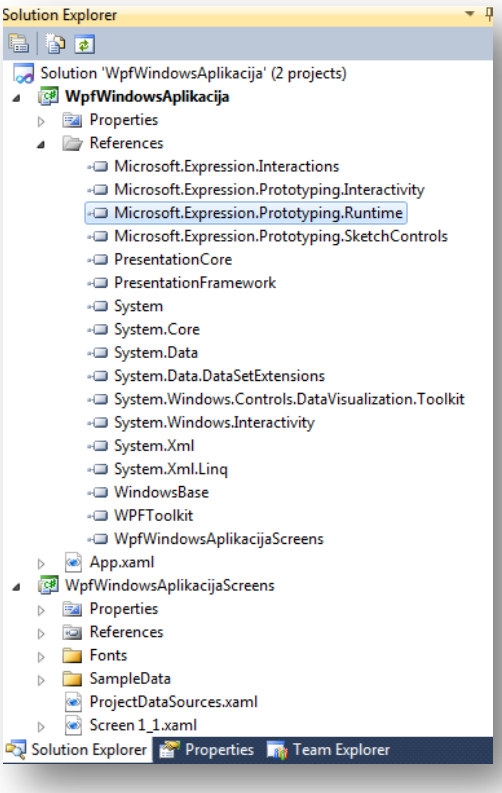
Slika 49: Microsoft Visual Studio 2010.

2. V Solution Explorer zavihku z desnim klikom kliknemo na »WpfWindowsAplikacija« in izberemo »Open Folder in Windows Explorer«. Odpre se nam raziskovalec, kjer imamo seznam datotek, ki jih vsebuje projekt.
3. V raziskovalcu poiščemo datoteko s končnico ».csproj« (WpfWindowsAplikacija.csproj) in odpremo to datoteko s programom, namenjenim oblikovanju besedila. Desni klik na datoteko in izberemo »Open with« oz. »Odpri z« in poiščemo program »Notepad«.
4. V odprti datoteki poiščemo in izbrišemo vrstice, ki so prikazane na spodnji sliki.

```
<ExpressionBlendPrototypingEnabled>false</ExpressionBlendPrototypingEnabled>
<ExpressionBlendPrototypeHarness>true</ExpressionBlendPrototypeHarness>
```

Slika 50: Koda za brisanje 1.

5. Shranimo in zapremo datoteko.
6. V Solution Explorer-ju poiščemo mapo »References«. V njej poiščemo »Microsoft.Expression.Prototyping.Runtime.dll« in ga izbrišemo.



Slika 51: Solution Explorer.

7. V Solution Explorer zavihku z desnim klikom kliknemo na »WpfWindowsAplikacijaScreens« in izberemo »Open Folder in Windows Explorer«. Odpre se nam raziskovalec, kjer imamo seznam datotek, ki jih vsebuje projekt.
8. V raziskovalcu poiščemo datoteko s končnico ».csproj« (WpfWindowsAplikacijaScreens.csproj) in odpremo to datoteko s programom, ki je namenjen oblikovanju besedila. Desni klik na datoteko in izberemo »Open with« oz. »Odpri z« in poiščemo program »Notepad«.
9. V odprti datoteki poiščemo in izbrišemo vrstice, ki so prikazane na spodnji sliki.

```
<ExpressionBlendPrototypingEnabled>false</ExpressionBlendPrototypingEnabled>
<ExpressionBlendPrototypeHarness>true</ExpressionBlendPrototypeHarness>
```

Slika 52: Koda za brisanje 2.

10. Shranimo in zapremo datoteko.
11. V Solution Explorer-ju poiščemo mapo »References«. V njej poiščemo »Microsoft.Expression.Prototyping.Runtime.dll« in ga izbrišemo.
12. V Solution Explorer poiščemo »App.xaml« in ga razširimo. Odpremo »App.xaml.cs«.
13. V »App.xaml.cs« poiščemo kodo, ki je prikazana na spodnji sliki.

```
[assembly: Microsoft.Expression.Prototyping.Services.SketchFlowLibraries("WpfWindowsAplikacija.Screens")]
```

Slika 53: Koda za brisanje 3

Ko najdemo to vrstico kode, si zapomnimo »WpfWindowsAplikacij.Screens« za kasnejšo rabo, nato izbrišemo to vrstico kode.

14. V »App.xaml.cs« poiščemo kodo, ki je prikazana na spodnji sliki.

```
this.StartupUri = new Uri(@"pack://application:,,,/Microsoft.Expression.Prototyping.Runtime;Component/WPF/Workspace/PlayerWindow.xaml");
```

Slika 54: Koda za brisanje 4.

Kodo, ki je prikazana na zgornji sliki, zamenjamo s kodo, ki je prikazana na spodnji sliki.

```
this.StartupUri = new Uri(@"pack://application:,,,/WpfWindowsAplikacija.Screens;Component/Screen 1_1.xaml");
```

Slika 55: Nova koda.

Ime »Screen1_1.xaml« je ime strani, ki se bo prikazala, ko bomo zagnali projekt.

15. Projekt zaženemo in odpre se nam WPF aplikacija (uspešno smo odstranili SketchFlow predvajalnik). [18]

Slika 56: WPF aplikacija.

5. Zaključek

Kakovost in čim krajši čas izdelave projekta sta bistvenega pomena tako za stranko kot za podjetje, ki se ukvarja z informacijskimi storitvami. Z izdelavo kakovostnih programov v najhitrejšem možnem času si podjetje širi dober glas in pridobiva stranke. To je še posebej pomembno v današnjem času recesije, ko stranke še toliko bolj pretehtajo denar in čas, ki ga bodo vložile v projekt. V teh primerih so orodja, kot je Microsoft Expression Blend 3, zelo dobrodošla, saj podjetju olajšajo delo in skrajšajo čas izdelave projekta Microsoft Expression Blend 3 omogoča oblikovalcem gradnjo privlačnih namiznih WPF ali spletnih Silverlight aplikacij in prototipov. Je vizualno in oblikovalcu prijazno orodje, ki omogoča risanje, animiranje, povezovanje z bazo podatkov, uvažanje slik, videov in 3D.

Za učinkovito izrabo zmožnosti tega orodja je sicer potrebno vložiti precej časa v učenje (povprečno 500 ur za študenta FRI), saj trenutno še ni na voljo veliko virov, ki bi se jih dalo uporabiti. Če pa želimo program uporabiti le za hitro predstavitev nekega prototipa, pa je čas učenja zelo majhen (povprečno 60 ur za študenta FRI). Preproste prototipe je namreč mogoče predstaviti v orodju le s klikanjem brez kodiranja.

V diplomski nalogi sem večji del predstavitve namenil spletni aplikaciji, saj tudi današnje tehnologije v vse večji meri težijo k spletnim aplikacijam v primerjavi z namiznimi. Dober primer predstavlja recimo prihajajoči Googlov operacijski sistem Google Chrome OS, ki bo spletni operacijski sistem. Za delovanje ne bo potreboval zmogljivega računalnika, pomnilnik bo deloval le kot trenutni pomnilnik. Vse se bo v bistvu odvijalo preko spleta.

Pri izdelavi prototipa sem imel težave pri nekaterih osnovnih stvareh, ki bi jih drugje rešil na enostaven način. Na drugi strani pa je bilo nekatere naloge tukaj lahko opraviti, medtem ko bi jih v drugih orodjih bilo težko izvesti. Res pa je, da sem se s tem orodjem prvič srečal pri pisanju diplomske naloge. V tem programu bi bilo dobro dopolniti možnost pretvorbe prototipa v dejansko aplikacijo s pomočjo čarovnika. Sedaj je potrebno spreminjati kodo, kar ni uporabniku prijazno.

Aplikacijo priporočam za uporabo, saj vedno znova prihajajo nove različice, ki vedno bolj poenostavljajo delo. S kontrolami, ki so na voljo, se da narediti marsikaj, potrebno je le znanje in veliko domišljije. Na Microsoftovih predavanjih, kjer so predstavljali beta verzijo Microsoft Expression Blend 4, so prikazali primer uporabe, kako je mogoče iz preprostega ListBox-a narediti skoraj nemogoče. ListBox je bil prikazan v obliki oblakov na modrem nebu, ki so bili različnih velikosti. Ti oblaki so se premikali in s klikom na njih je začel padati dež. Po določenem času pa so oblaki izginili. Posebej bi želel še poudariti, da je v Expression Blend-u edina omejitev znanje in domišljija. Vse je mogoče. Potreben je le pogum, da začnete z učenjem tega izjemnega orodja.

Kazalo slik

SLIKA 1: SHEMA VIZIJE 6
SLIKA 2: SHEMA PLANIRANJA 11
SLIKA 3: MODELI NAČRTOVANJA 13
SLIKA 4: TRIJE NIVOJI STORITEV 18
SLIKA 5: SHEMA RAZVOJA 23
SLIKA 6: RAZLIČICE PRODUKTA 25
SLIKA 7: SHEMA STABILIZACIJE 28
SLIKA 8: EXPRESSION BLEND. 33
SLIKA 9: DELOVNO OKOLJE PRED POPRAVKI 35
SLIKA 10: DELOVNO OKOLJE PO POPRAVKIH. 35
SLIKA 11: ZAVIHEK OBDELOVALNEGA OKNA 36
SLIKA 12: ZAVIHEK SKETCHFLOW MAP. 36
SLIKA 13: ZAVIHEK PROJECTS. 37
SLIKA 14: ZAVIHEK ASSETS. 37
SLIKA 15: ZAVIHEK OBJECTS AND TIMELINE. 38
SLIKA 16: ZAVIHEK PROPERTIES. 38
SLIKA 17: ZAVIHEK SKETCHFLOW ANIMATION. 38
SLIKA 18: PREDVAJALNIK SKETCHFLOW PLAYER. 39
SLIKA 19: NAVIGACIJSKO OKNO V PREDVAJALNIKU SKETCHFLOW PLAYER. 39
SLIKA 20: FEEDBACK V SKETCHFLOW PLAYERJU. 40
SLIKA 21: SKETCHFLOW MAP IN NODI. 40
SLIKA 22: KONTROLE, KI SO TRENUTNO NA IZBRANEM OKNU. 41
SLIKA 23: OKNO FILMANIJA. 41
SLIKA 24: IKONA KOMPONETNEGA OKNA NAVBAR. 42
SLIKA 25: KOMPONETNO OKNO NAVBAR NA OKNU FILMANIJA. 42
SLIKA 26: PRIKAZNO OKNO ZA IZBIRO XML PODATKOV. 43
SLIKA 27: PRIKAZNO OKNO ZA IZBIRO TOČNO DOLOČENIH PODATKOV. 43
SLIKA 28: OKNO OPIS, KI PRIKAZUJE OPIS FILMA. 44
SLIKA 29: KONTROLE, KI BOJO UPORABLJENE ZA ANIMACIJO. 44
SLIKA 30: KONTROLE, KI SO TRENUTNO NA IZBRANEM OKNU. 45
SLIKA 31: PRVA STVAR, KI JO NAREDIMO PRI ANIMACIJI. 45
SLIKA 32: PRIKAZ SLIKE PO KLIKU NA IKONO PLUSA. 45
SLIKA 33: PRIKAZ PREMIKA PRVEGA DELA ANIMACIJE. 46
SLIKA 34: PRIKAZ REZULTATA ISKANJA. 46
SLIKA 35: PRIKAZ OPISA FILMA. 47
SLIKA 36: PRIKAZ TRAILERJA. 47
SLIKA 37: PRIKAZANO OKNO ZA IZVOZ PROJEKTA V WORD DOKUMENT. 49
SLIKA 38: SKETCHFLOW PLAYER. 49
SLIKA 39: SKECTFLOW PLAYER MAP. 50
SLIKA 40: PRIKAZ NAPAKE 1. 50
SLIKA 41: PRIKAZ NAPAKE 2. 51
SLIKA 42: FEEDBACK ORODJA. 51
SLIKA 43: UVOŽENI FEEDBACK 52
SLIKA 44: SPLETNA APLIKACIJA - OSNOVNA STRAN (SEZNAM FILMOV). 53
SLIKA 45: ISKANJE FILMA V ZAVIHKU FILMI. 54
SLIKA 46: OPIS FILMA ODRASLI. 54
SLIKA 47: PRIKAZ TRAILERJA FILMA ODRASLI. 55
SLIKA 48: WPF PROTOTIP IN APLIKACIJA PRED PRETVORBO. 56

SLIKA 49: MICROSOFT VISUAL STUDIO 2010. 57
SLIKA 50: KODA ZA BRISANJE 1. 57
SLIKA 51: SOLUTION EXPLORER. 58
SLIKA 52: KODA ZA BRISANJE 2. 58
SLIKA 55: NOVA KODA. 59
SLIKA 56: WPF APLIKACIJA. 59
SLIKA 53: KODA ZA BRISANJE 3. 59
SLIKA 54: KODA ZA BRISANJE 4. 59

Kazalo tabel

TABELA 1: TIPIČNE ZADOLŽITVE VLOG V FAZI VIZIJE 7
TABELA 2: TIPIČNE ZADOLŽITVE VLOG V FAZI PLANIRANJA 12
TABELA 3: MODELI NAČRTOVANJA 12
TABELA 4: FUNKCIONALNE SPECIFIKACIJE 21
TABELA 5: VLOGE IN NJIHOVE ZADOLŽITVE 21
TABELA 6: VSEBINA PROJEKTNEGA PLANA 22
TABELA 7: TIPIČNE ZADOLŽITVE V FAZI RAZVOJA 24
TABELA 8: TIPIČNE ZADOLŽITVE VLOG V FAZI STABILIZACIJE 29

Viri in literatura

[1] Microsoft Corporation, Analyzing Requirements and Defining Microsoft® .NET Solution Architectures, Microsoft Press, 2003

[2] Scott Wilson, Bruce Maples, Tim Landgrave, Analyzing Requirements and Defining Solution Architectures, Kizan Corporation, 1999

[3] (2010) Osnove projektnega vodenja. Dostopno na: http://www.ra-sinergija.si/projektno_vodenje/1__uvod.html

[4] (2010) Wikipedia - WPF, Silverlight in XAML. Dostopno na: http://sl.wikipedia.org/wiki/Windows_Presentation_Fundation_vs._Windows_Forms

[5] (2010) Wikipedia - Kaj je projekt. Dostopno na: http://sl.wikipedia.org/wiki/Projekt

[6] (2002) MSF Process Model v. 3.1. Dostopno na: http://download.microsoft.com/download/7/7/7/777104c9-506e-47c9-9da4-9e23138be493/MSF%20Process%20Model%20v.%203.1.pdf

[7] (2006) MSF Process model cookbook. Dostopno na: http://developers.de/media/p/436/download.aspx

[8] (2010) Microsoft Visio. Dostopno na: http://office.microsoft.com/sl-si/visio/

[9] (2009) Alternative Microsoft Visio. Dostopno na: http://www.osalt.com/visio

[10] (2010) Opis Microsoft Expression Blend. Dostopno na: http://www.microsoft.com/slovenija/msdn/expression/expression-blend.mspx

[11] (2010) Opis Microsoft Project. Dostopno na: http://www.b2.eu/racunalniski-tecaji/projektno-vodenje.aspx

[12] (2009) Alternative Microsoft Project. Dostopno na: http://www.osalt.com/project

[13] (2010) Opis Microsoft Visual Studio. Dostopno na: http://www.reproms.si/prodaja/razvojna_orodja.wlgt

[14] (2009) Alternative Expression Blend. Dostopno na: http://www.osalt.com/visual-studio

[15] (2009) Nekaj o Microsoft Expression Blend 3. Dostopno na: http://www.msblogs.si/post/Primeri-uporabe-Microsoft-Expression-Blend-3.aspx

[16] (2010) Opis Microsoft Expression Blend 4. Dostopno na: http://www.microsoft.com/expression/products/Blend_Features.aspx

[17] (2010) Video primeri Expression Blend 3. Dostopno na: http://expression.microsoft.com/en-us/cc197141.aspx

[18] (2010) Kako pretvoriti WPF aplikacijo z uporabo Visual C#. Dostopno na: http://msdn.microsoft.com/en-us/library/ee371158%28Expression.30%29.aspx

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Jamnik

Razvoj prototipov v aplikaciji z orodjem Microsoft Expression Blend 3

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Matjaž Kukar

Ljubljana, 2010

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

Spodaj podpisani **Anže Jamnik**,

z vpisno številko **63040217**,

sem avtor diplomskega dela z naslovom:

Razvoj prototipov aplikacij z orodjem Microsoft Expression Blend 3

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne: 15.09.2010 Podpis avtorja:

Zahvala

Zahvalil bi se doc. dr. Matjažu Kukarju za strokovno pomoč in vodenje pri izdelavi diplomske naloge. Zahvalil bi se tudi mojemu šefu v podjetju B2 d.o.o., ker mi je pomagal izbrati temo za diplomsko nalogo in priskrbel literaturo. Zahvalil bi se tudi vsem, ki so kakorkoli pomagali pri diplomski nalogi. Posebna zahvala pa gre moji družini, ki mi je omogočila študij in me vseskozi podpirala.

Kazalo

Povzetek 1
Abstract 2
1. Uvod 3
2. Osnove projektnega dela 4
2.1. Splošno 4
2.2. Izhodišča za delo 4
2.3. Financiranje dejavnosti razvoja 5
2.4. Vloge 5
3. Štiri faze projekta 6
3.1. Vizija 6
3.1.1. Splošno 6
3.1.2. Kdo jo izdela? 7
3.1.3. Metode dela 7
3.1.4. Komunikacija in Expression Blend 3 8
3.1.5. Mejnik odobritev vizije ter izdelki faze vizije 8
3.2. Planiranje 10
3.2.1. Splošno 10
3.2.2. Kdo ga izvede? 11
3.2.3. Metode dela 12
3.2.4. Konceptualno načrtovanje 13
3.2.5. Logično načrtovanje 15
3.2.6. Fizično načrtovanje 15
3.2.7. Mejnik odobritev projektnega plana in izdelki faze planiranja 20
3.3. Razvoj 22
3.3.1. Splošno 22
3.3.2. Kdo ga izvede? 23
3.3.3. Metode dela 24
3.3.4. Implementacija 24
3.3.5. Mejnik zaključen razvoj ter izdelki faze razvoja 25
3.4. Stabilizacija 27
3.4.1. Splošno 27
3.4.2. Kdo jo izvede? 28
3.4.3. Metode dela 29
3.4.4. Mejnik izdaja različice ter izdelki faze stabilizacije 30
3.5. Dostava produkta 31
3.6. Programska orodja, ki pomagajo pri razvoju aplikacij 31
3.6.1. Microsoft Visio 31
3.6.2. Microsoft Expression Blend 32
3.6.3. Microsoft Project 32
3.6.4. Microsoft Visual Studio 32
4. Expression Blend 3 – enostavna rešitev za izdelavo prototipov 33
4.1. Expression Blend 4 in njegove novosti 34
4.2. Prvi stik z novim okoljem in ureditev okolja na delo 34
4.2.1. Predstavitev zavihkov 36
4.3. Predstavitev SketchFlow player-ja 39
4.4. Izdelava prototipa preproste spletne aplikacije 40
4.4.1. Kratek opis aplikacije 40
4.4.2. SketchFlow map in nodi 40

4.4.3. Komponentna okna 41
4.4.4. Kako do komponentnega okna 42
4.4.5. Gumbi in akcije 42
4.4.6. Iskalnik in xml podatki 42
4.4.7. Okno Opis in primer opisa filma 43
4.4.8. Animacija iskanja in izbira filma 44
4.4.9. Predloga 47
4.4.10. Izvoz projekta in povratne informacije 48
4.4.11. Povratne informacije (Feedback) 49
4.5. Preprosta spletna aplikacija 53
4.5.1. Slike končne aplikacije 53
4.6. Preprosta WPF prototipna namizna aplikacija 56
4.7. Pretvorba WPF prototipa v WPF aplikacijo. 56
4.7.1. Prikaz pretvorbe WPF prototipa v WPF aplikacijo 56
5. Zaključek 60
Kazalo slik 61
Kazalo tabel 63
Viri in literatura 64

Seznam kratic in simbolov

API	Application programming interface
MS PowerPoint	Microsoft PowerPoint
MSF	Microsoft Solutions Framework
RHB	Različica brez hroščev
SQL	Structured Query Language
SUBP	Sistem za upravljanje baze podatkov
TFS	Team Foundation Server
UML	Unified Modeling Language
UPW	Unified Process Workflows
VS2010	Microsoft Visual Studio 2010
WPF	Windows Presentation Framework
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
ZBR	Zero Bug Release

Povzetek

V diplomski nalogi sem se osredotočil na prikaz poteka izdelave aplikacije. Predstavil sem vse faze projekta, ki so potrebne za dosego le tega. Predstavil sem tudi načine, kako je mogoče olajšati vloženo delo ter skrajšati čas, ki je potreben za zaključek nekega projekta. V ta namen sem uporabil orodje Microsoft Expression Blend 3. S pomočjo tega orodja lahko izdelujemo prototipe aplikacij in jih istočasno uporabimo tudi pri dejanski aplikaciji. V aplikaciji je dobro poskrbljeno tudi za komunikacijo. Stranka in razvijalec si lahko izmenjujeta informacije na enostaven način.

V diplomski nalogi sem predstavil orodje Expression Blend 3 ter izdelavo enostavnega prototipa s tem orodjem. Prototip sem naredil s pomočjo tehnologije Silverlight, ki je namenjena spletnim aplikacijam. Celoten prototip je narejen brez vnašanja kode. V okviru prototipa sem predstavil potek komunikacije. Sprogramiral sem še končno aplikacijo v okolju Microsoft Visual Studio 2010 s pomočjo C# in .NET. Tudi izgled končne aplikacije je prikazan v diplomi. Aplikacija ima v ozadju bazo podatkov (SQL Server 2008). Predstavil sem še preprosto namizno aplikacijo, ki temelji na WPF tehnologiji. Pri namizni aplikaciji sem prikazal, kako se prototip aplikacije pretvori v dejansko aplikacijo. Po pretvorbi se je ohranilo vse.

Ključne besede: Microsoft Expression Blend 3, Microsoft Visual Studio 2010, SQL Server 2008, C#, .NET, prototip, aplikacija, komunikacija, Silverlight, WPF, XAML.

Abstract

The BA thesis focuses on the process of application building. All project phases required to build an application as well as methods for facilitating the work and shortening the time needed to complete a certain project are presented. With the help of Microsoft Expression Blend 3 the application prototypes can be developed and used with an actual application at the same time. The exchange of information between the customer and the developer is simple as good communication is a key factor in the application.

The BA thesis also presents the Expression Blend 3 and the development of a simple prototype using this tool. The prototype was developed using the Silverlight technology which is designed for web applications. The entire prototype was developed without writing any code. The process of communication is presented within the framework of the prototype. The final application was programmed in Microsoft Visual Studio 2010 environment using C# and .NET. The appearance of the final application is also shown. The application is using a SQL Server 2008 database. The BA thesis presents a simple desktop application based on the WPF technology and the process of transformation from a prototype application to an actual application. After the transformation there was no change.

Keywords: Microsoft Expression Blend 3, Microsoft Visual Studio 2010, SQL Server 2008, C#, .NET, prototype, application, communication, Silverlight, WPF, XAML.

1. Uvod

Pisanje projektov ni muha enodnevnica in je za marsikaterega posameznika ali za podjetje velikega pomena. Vključuje veliko znanj in veščin, ki si sproti pridobivamo. Vsa podjetja razpolagajo z omejenim številom virov, kot so ljudje, material in stroški. Projektni vodje imajo omejeno število virov, ki jih lahko uporabijo na projektih. Zato je pomembno, da je vsaka faza projekta čim hitreje in čim bolj kakovostno zaključena. Problematičen del vsakega projekta je komunikacija med strankami in razvijalci. Težko se je sporazumeti, kakšne so želje stranke. Komunikacija in izdelava prototipov predstavlja velik problem, tako da sem se v diplomski nalogi lotil raziskovanja tega področja. [3]
Glavni cilj diplome je prikazati potek projekta od ideje do končne delujoče aplikacije in pa predstaviti orodje Expression Blend 3. S tem orodjem lahko enostavno gradimo prototipe aplikacij in rešujemo problem komunikacije. Prikazal bom potek izdelava prototipa spletne aplikacije. V okviru spletne aplikacije bom predstavil tudi način komunikacije v tem orodju. Predstavil bom tudi primer namiznega prototipa, pri katerem bom prikazal postopek pretvorbe prototipa v aplikacijo.
V drugem in tretjem poglavju je bralcu predstavljen projekt na splošno ter faze projekta, ki so potrebne za uspešno vodenje projekta. Znotraj posameznih faz projekta je natančno predstavljen namen faze, kdo jo izdela, katere metode dela so potrebne, na kak način poteka komunikacija in kakšni so mejniki za odobritev faze.
Četrto poglavje in glavni del diplomskega dela je namenjen predstavitvi orodja Expression Blend 3 in izdelavi prototipov. Expression Blend ima na voljo dve platformi. Ti dve platformi sta WPF in Silverlight. Za kodiranje pa imamo na voljo dva .NET jezika, C# in Visual Basic, ter opisni programski jezik XAML za dizajn.
WPF in Silverlight sta platformi, ki sta uporabni v zahtevnih aplikacijah z uporabo 3D, interaktivnosti, multimedije in prilagojenimi uporabniškimi vmesniki. Trenutno najboljša kombinacija za hiter razvoj WPF in Silverlight aplikacij je Microsoft Visual Studio 2010 in Microsoft Blend 4.
Iz tega je vidna težnja Microsofta po ločenosti oblike programa, ki jo naredi oblikovalec v Expression Blend 4, od kode oziroma poslovne logike programa, ki jo spiše programer v Visual Studiu. Prednost teh dveh platform je tudi sam opisni jezik XAML, ki je narejen po standardih XML jezika. S tem lažje ločimo oblikovni del aplikacije od programerskega dela, seveda pa kasneje lažje uporabimo že napisano XAML kodo (ti. predlogo). Ker so fizično ločene tudi datoteke, lahko oblikovalec in programer delata istočasno na istih datotekah. Tu je dobro omeniti povezljivost z Silverlight aplikacijami, saj si WPF in Silverlight delita iste osnove, npr. XAML. WPF in Windows Forme sta tudi popolnoma kompatibilna, saj lahko v Windows Forme vstavljamo elemente WPF in obratno.
XAML je osnova za WPF in Silverlight grafiko. Uporablja se za vektorske slike, stile, barvne palete, grafične vmesnike, ustvarjanje dokumentov za tisk in še mnogo drugih stvari. [4]

2. Osnove projektnega dela

2.1. Splošno

Projekt je enkratna, praviloma zahtevna in kompleksna skupina nalog, ki mora biti končana v določenem roku, doseči mora vnaprej določene in morebitne kasneje odkrite cilje, ter upoštevati vse podane in kasneje odkrite omejitve.
Projekt je ciljno usmerjen in zaključen proces razvijanja dejavnosti, ki so usmerjene k doseganju končnega cilja. Do tega cilja se prihaja postopoma z doseganjem posameznih podciljev. Pobuda za projekt lahko vsebuje samo en končni cilj ali pa vse podcilje. Pobudnik projekta je lahko posameznik, podjetje, družbena organizacija, javna organizacija, družbena institucija, državna institucija ali celo mednarodna organizacija, ki je običajno tudi naročnik projekta. Naročnik projekta običajno postavi tudi cilje projekta. Za uspešno realizacijo projekta je potrebno poiskati še izvajalce in določiti vodje projekta. Vodstvo ima pri organiziranju, upravljanju in vodenju projekta odločilno vlogo. Vodstvo projekta mora načrtovati in aktivirati posamezne aktivnosti ter jih tehnično, časovno in finančno uskladiti. Projekti so lahko majhni ali veliki, vsi pa morajo biti časovno omejeni, predviden mora biti čas začetka in zaključka. Uspešno izvajanja projekta zahteva tudi dobro postavljen informacijski sistem. [5]
Za planiranje in vodenje projektov si pomagamo z razvojno procesnim modelom MSF (Microsoft Solution Framework).

2.2. Izhodišča za delo

Osnovno izhodišče za aktivnosti na področju razvoja je projekt. Vsaka aktivnost, ki se izvaja na področju razvoja, tako pripada določenemu projektu. Projekt se lahko oblikuje na podlagi:

- *Prodaje (zahteva po pripravi ponudbe, zahteva po pripravi analize, …),*
- *stranke in*
- *vodje razvoja na podlagi plana razvoja.*

V sklopu vsakega projekta se na področju razvoja izvajajo naslednje faze:

- *Vizija,*
- *planiranje,*
- *razvoj in*
- *stabilizacija.*

Posamezno nalogo v določeni fazi lahko odredijo naslednje vloge v projektu:

- *Produktno upravljanje: vizija, načrtovanje,*
- *razvoj: razvoj,*
- *projektno upravljanje: načrtovanje, razvoj, stabilizacija,*
- *testiranje: stabilizacija,*
- *vzdrževanje: stabilizacija.*

Posamezne naloge se izvajajo po pravilih, opisanih v poglavjih v nadaljevanju in v skladu s Shemo razvoja. [2]

2.3. Financiranje dejavnosti razvoja

Dejavnosti razvoja se lahko financirajo s prodajo rešitev ter iz rezerv oddelka IS.
Viri financiranja so:

- kupnine od prodanih programov,
- plačljiva dela, opravljena za stranke, in
- vzdrževalne pogodbe. [2]

2.4. Vloge

Za vsak projekt se vzpostavi projektna skupina, ki je razdeljena v naslednje vloge:

- *Produktno upravljanje* (kupčev zagovornik v timu in obratno, skrb za produktne dokumente, zmogljivosti produkta, prioritete razvoja ter komunikacijo),
- *projektno upravljanje* (odgovornost za dostavo pravega produkta ob pravem času, skrb za pravi obseg produkta tudi v finančnem smislu),
- *razvoj* (izdelava in testiranje, ocena časa in napora za realizacijo, svetovanje),
- *testiranje* (določitev strategije testiranja, testiranje, preverjanje kvalitete) in
- *vzdrževanje* (določitev strategije namestitve, izvedba namestitve, upravljanje z nadgradnjami).

Na manjših projektih (manj od 500 ur) sta vlogi *produktno* in *projektno upravljanje* združeni (kot vloga *vodja projekta*). Prav tako ni nujno, da obstaja vloga *vzdrževanja*, ampak so v tej vlogi lahko člani iz vlog *razvoj* in *testiranje*. Ne glede na velikost projekta pa ni dopustno, da zgolj člani vloge *razvoj* izvajajo naloge testiranja. [2]

3. Štiri faze projekta

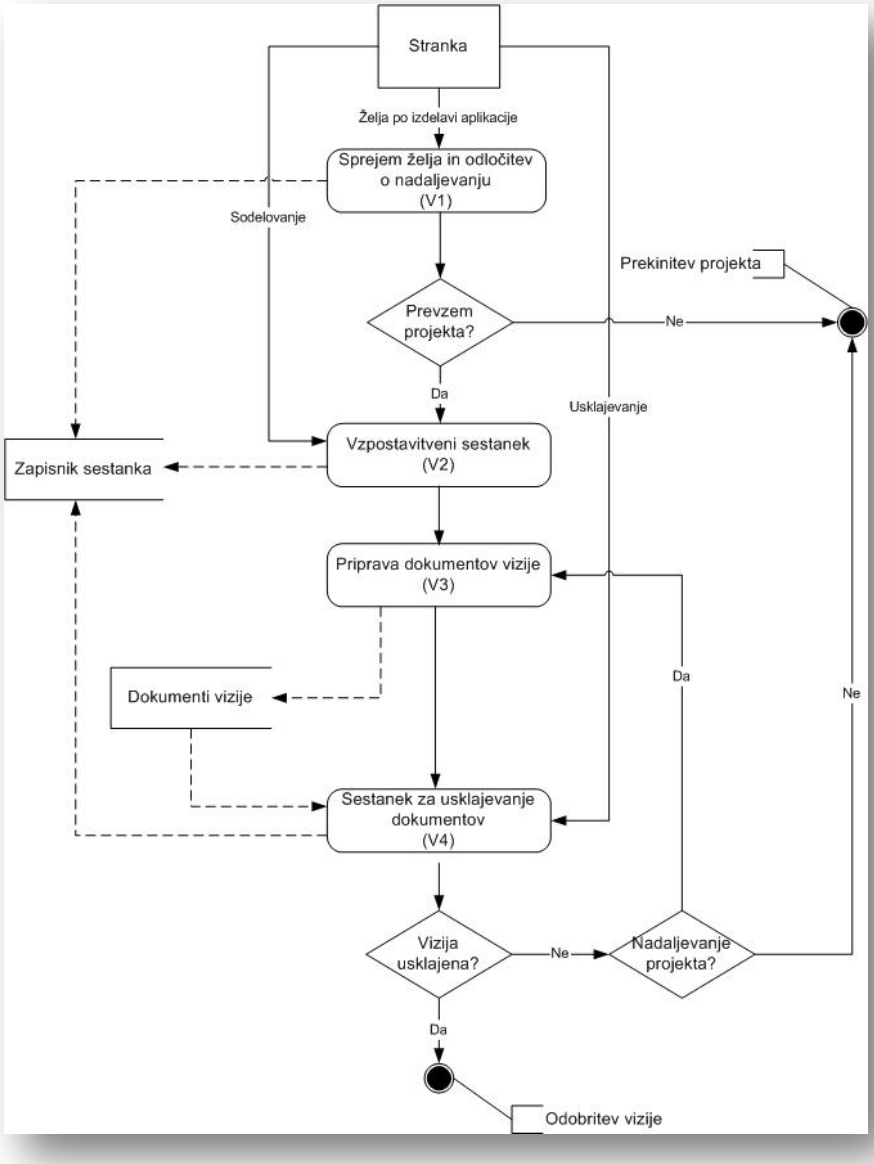
Projektne faze delimo na štiri dele:

- *Vizija,*
- *planiranje,*
- *razvoj in*
- *stabilizacija.*

3.1. Vizija

3.1.1. Splošno

Pred začetkom kateregakoli dela projekta mora tim izdelati *vizijo*, ki opisuje skupni cilj in smer projekta. Med samo izdelavo vizije mora imeti tim svobodo za »brainstorm« vseh elementov projekta. Priprava in usklajevanje dokumenta vizije omogoča timu in stranki zbistritev ciljev projekta in omogoči, da vsi vidijo, kaj je namen zaključenega projekta.
Vizija predstavlja približno 10 do 15% celotnega projekta. [2]



Slika 1: Shema vizije

3.1.2. Kdo jo izdela?

Spodnja tabela opisuje tipične zadolžitve vlog v fazi *vizije*. Vodja vsake vloge skrbi za izvršitev zadanih nalog in komunikacijo z ostalimi člani tima.

Tabela 1: Tipične zadolžitve vlog v fazi vizije

Vloga	Zadolžitve
Produktno upravljanje	Dostavi dokument vizije. Upravlja zahteve stranke. Vključuje stranko pri prototipnem razvoju. Upravlja z tveganji projekta
Projektno upravljanje	Izdela cilje dizajna. Opiše koncept rešitve. Določi okvirje projektne strukture. Upravlja s tveganji projekta
Razvoj	Načrtuje prototipe. Opiše okvire možnosti razvoja. Identificira posledice.
Uvajanje	Identificira uporabniške zahteve po učinkovitosti in posledice. Upravlja s pričakovanji uporabnikov. Vključuje uporabnike pri prototipnem razvoju. Upravlja s tveganji projekta.
Testiranje	Specificira kriterije odobritve. Skicira sistem za odkrivanje hroščev. Skicira sistem za upravljanje s tveganji. Identificira tveganja projekta.
Logistično upravljanje	Identificira posledice razvoja. Identificira posledice podpore. Upravlja s tveganji projekta.

[6]

3.1.3. Metode dela

Pri pripravi vizije projekta se uporabljajo koraki, podobni UPW (Unified Process Workflows). Sledijo si v fleksibilnem zaporedju, ki omogoča vračanje v prejšnje korake. Koraki so: *raziskava, analiza, implementacija* in *validacija*. [2]

3.1.3.1. Raziskava

Za izdelavo vizije mora tim opraviti določene raziskave, kjer se osredotoči na stranko in na druge podobne produkte, razvite ali uporabljene s strani organizacije.

Cilj in izhod: Raziskati in zabeležiti je potrebno čim več informacij. Izvori teh informacij naj bodo stranka, uporabniki, trenutne aplikacije in njihova dokumentacija kakor tudi strokovnjaki iz obravnavanega področja. [2]

Raziskava drugih aplikacij podjetja

Če že obstaja kakšna predhodna različica produkta, naj se tim najprej seznani z vizijo predhodne različice. Velika verjetnost je, da predhodna različica vsebuje dolgoročne cilje, ki naj bi jih naslednje različice implementirale. Podobno pa mora tim razumeti tudi vizije drugih aplikacij, ki so trenutno v razvoju. Nekatera vprašanja, ki naj si jih tim zastavi ob pripravi vizije za nov produkt, so:

- Ali predstavlja nov produkt velik ali majhen korak za organizacijo?
- Ali sta zamišljen koncept in arhitektura nova za organizacijo?
- Katere novosti prinaša produkt za organizacijo? [2]

Raziskava stranke in uporabnikov

Faza Vizije predstavlja čas, ko se tim osredotoči na problematiko stranke in kako bodo uporabniki produkt uporabljali da bodo le to reševali. Področja, ki naj jih tim pregleda, so: poslovne pobude, obstoječe informacije, nova strojna oprema, nove tehnologije, problematika vzdrževanja, plani uvajanja uporabnikov, globalna problematika. [2]

Raziskava podobnih produktov
Vizija, ki jo tim izdela za produkt, mora biti usklajena z vizijami za že razvite produkte organizacije. Vprašanja, ki naj si jih tim ob tem zastavi, so naslednja: vpliv vizij ostalih produktov, vpliv operacijskega sistema na obstoječe produkte, urnik razvoja ostalih produktov za organizacijo v trenutnem obdobju. [2]

3.1.3.2. Analiza
Cilj: V *fazi analize* tim analizira obstoječe stanje s pomočjo informacij o poslovanju in informacij od uporabnikov. Ugotoviti je potrebno trenutne značilnosti.
Izhod: Izhodišče predstavljajo diagrami primerov uporabe in diagrami aktivnosti, pripravljeni po UML, ki najlažje predstavijo poslovne in uporabniške zahteve. V dodatno pomoč so tudi opisi specifičnih primerov uporabe, scenarijev primerov uporabe in diagramov aktivnosti. Ti diagrami predstavljajo osnovo za celotni proces razvoja. [2]

3.1.3.3. Racionalizacija
Po izdelanih primerih uporabe in seznamu značilnosti je potrebno podobne značilnosti združiti v večje sklope na podlagi uporabniških akcij, akcij aplikacij, uporabljenih podatkov ali drugih smiselnih grupiranj. V tej fazi se tim še vedno ukvarja z značilnostmi, ki opisujejo trenutno stanje, kakor tudi že značilnostmi bodočega stanja.
Cilj: Racionalizirati je potrebno izdelane modele v želji po doseganju večje učinkovitosti aktivnosti.
Izhod: Izhodišče predstavljajo racionalizirani primeri uporabe, njihovi scenariji in diagrami aktivnosti. [2]

3.1.3.4. Implementacija
V *fazi implementacije* tim pregleda predhodno definirane sklope značilnosti in jim določi prioritete. Skupaj so zbrani tudi primeri uporabe, njihovi scenariji in diagrami uporabe bodočega stanja, izdelan pa je tudi osnutek dokumenta vizije. Ta vsebuje dolgoročne cilje projekta. V tem trenutku je znanih dovolj podatkov, da je lahko izdelan, optimiziran in sprejet trikotnik variabilnosti virov, značilnosti in datuma zaključka. Prav tako je lahko izdelan okvirni projektni plan, ki se uporabi kot osnova za *fazo planiranja*.
Cilj: Izdelati je potrebno osnutek dokumenta vizije in projektnega plana.
Izhod: Izhodišče je osnutek dokumenta vizije in projektnega plana. [2]

3.1.3.5. Validacija
Cilj: Cilj validacije je revizija in dodelava dokumenta vizije. Prav tako se je potrebno ozreti na predhodne korake in izdelke ter ugotoviti pripravljenost tima na mejnik *vizija odobrena*.
Izhod: Rezultat validacije je revidiran in usklajen končni dokument vizije. [2]

3.1.4. Komunikacija in Expression Blend 3
Faza vizije ni samo zbiranje in priprava dokumentov, temveč odprta debata o viziji projekta projektnega tima in predstavnikov stranke. Prototipi predstavljajo učinkovit način komunikacije. V Expression Blend-u 3 imamo hiter in učinkovit sistem za komunikacijo. Z njim se lahko izognemo številnim sestankov. Več o tem je razloženo v poglavju, ki opisuje Expression Blend. [2]

3.1.5. Mejnik odobritev vizije ter izdelki faze vizije
Cilj *faze vizije* je doseg mejnika *odobritev vizije*, ki predstavlja dogovor oziroma pogodbo med stranko / naročnikom in timom o glavnih kritičnih izhodiščih projekta.
Za doseg mejnika *odobritev vizije* so potrebni naslednji izdelki:

- dokument vizije,
- dokument projektne skupine in
- glavni dokument tveganj.

Na tem mejniku se tim tudi odloči o nadaljevanju projekta ali njegovi opustitvi. Kajti dobro zaključena *faza vizije* lahko timu in stranki oziroma naročniku omogoča nadaljevanje projekta brez strahu pred neuspehom. [2]

3.1.5.1. Dokument vizije

Dokument vizije, ki je izdelan ob koncu *faze vizije*, je učinkovit, če vsebuje vsaj naslednje elemente:

- izjavo vizije,
- raziskavo uporabnikov,
- informacije o konkurenci,
- bodoče sklope značilnosti in
- grob projektni urnik. [1]

Definiranje izjave vizije

Izjava vizije predstavlja veliko lastnosti pametnega cilja, ki so:

- specifičnost (izrecen, svojstven),
- merljivost (lahko v vsakem trenutku izmerimo, kolikšen je napredek),
- izvedljivost,
- relevantnost in
- časovna definiranost.

Pravilo za jedrnato izjavo vizije je enostavno definirano in vsem razumljivo. [1]

Raziskava uporabnikov

Vse, kar tim z raziskavo pridobi od stranke in uporabnikov, predstavlja osnovo za nadaljnje korake. To upošteva raziskavo konteksta, raziskavo trga, ciljne skupine, … Raziskava tudi ponudi informacije za pripravo primerov uporabe, ki opisujejo trenutno in bodoče delo uporabnikov. [1]

Informacije o konkurenci

Tu si zastavimo vprašanje, katere druge aplikacije, ki ponujajo rešitev zahtev stranke, so še na trgu. Tim mora zagotoviti, da produkt, opisan z vizijo, rešuje zahteve stranke bolje kakor katerakoli druga obstoječa aplikacija in pri tem še upravičuje investicijo. [1]

Opis bodočih sklopov značilnosti

Sklopi značilnosti so kategorije, v katero bodo spadale vse bodoče značilnosti produkta. Programsko upravljanje uporabi te kategorije pri ugotavljanju, ali bodoče značilnosti zadovoljujejo cilje projekta. Če značilnost ne spada v noben sklop, najbrž ni pomembna za trenutno različico. [1]

Določitev prioritet in projektnega urnika

Po zaključku *faz raziskave*, *analize* in *racionalizacije* v *procesu vizije* ima tim grob načrt projektnega urnika in virov za njegovo uresničitev. Tim izdela ustrezen trikotnik variabilnosti, ki prikazuje porazdelitev virov, značilnosti in datum zaključka. [1]

3.1.5.2. Dokument projektne skupine

Dokument za razliko od dokumenta vizije, ki opisuje, kaj naj bo izdelano, opisuje, kdo naj to izdela. Dokument opisuje:

- vse vloge tima,
- kdo je vodja posamezne vloge,
- kdo so člani tima in njihove kontaktne informacije,
- kdo je naročnik projekta,
- kako bo projekt upravljan, npr. kontrola nad spremembami in način poročanja,
- urnik sestankov projekta, e-pošta in spletne strani. [1]

3.1.5.3. Glavni dokument tveganj

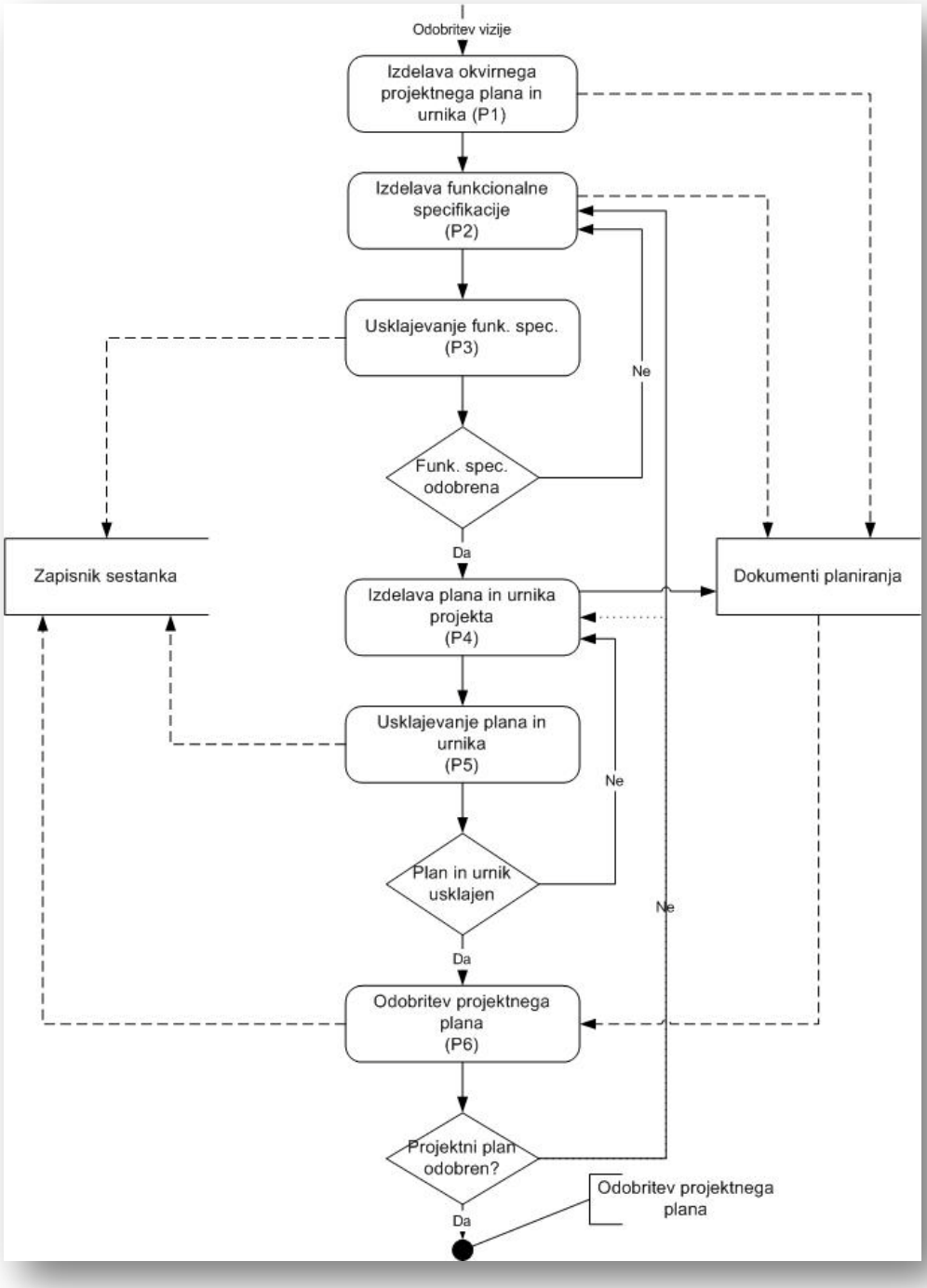
Dokument vsebuje analizo tveganj projekta ter plana za njihovo odpravo. Večji del dokumenta predstavlja spisek tveganj z opisi. Ta so največkrat predstavljena kot najpomembnejših deset tveganj. Plani odprave pa vsebujejo način odprave ter kdo je zadolžen za njihovo izvršitev. Prav tako je priporočljiv graf, ki predstavlja tveganja, njihovo verjetnost pojavitve in vpliva na projekt. [1]

3.2. Planiranje

3.2.1. Splošno

Medtem, ko je bilo v *fazi vizije* glavno vprašanje »Ali lahko s pomočjo tehnologije rešimo določen poslovni problem in če, kako?« in rezultat razumevanje problema in skupna vizija rešitve, je v *fazi planiranja* glavno vprašanje »Kaj v resnici potrebujemo, da dosežemo zastavljeno vizijo?«. Rezultat pa je podroben projektni plan, ki ga odobravata tako stranka kot tudi projektni tim. Tu se tudi zastavijo najpomembnejša vprašanja, ki so: »Katere značilnosti bodo počakale na naslednjo različico?«, »Katera tehnologija še ni zrela za uporabo?«, »Kateri so tisti stroški, ki ne upravičujejo razvoja?«, …

Planiranje predstavlja približno med 35 do 40% celotnega projekta. [2]



Slika 2: Shema planiranja

3.2.2. Kdo ga izvede?

Spodnja tabela opisuje tipične zadolžitve vlog v *fazi planiranja*. Vodja vsake vloge skrbi za izvršitev zadanih nalog in komunikacijo z ostalim člani tima.

Tabela 2: Tipične zadolžitve vlog v fazi planiranja

Vloga	Zadolžitve
Produktno upravljanje	Vodi proces zbiranja zahtev in proces konceptualnega načrtovanja. Dela na planu in urniku komunikacij.
Projektno upravljanje	Vodi celoten proces načrtovanja, predvsem logično načrtovanje. Izdela osnutek funkcionalne specifikacije. Prevzame vodstvo nad celoto in preverja, ali tim dosega plan in urnik.

Razvoj	Vodi fizični del načrtovanja funkcionalne specifikacije. Določi čas in obremenitev za izgradnjo in stabilizacijo produkta, ki ga specificira v planu in urniku razvoja. Razvije potrebne sisteme preverjanja ustreznosti koncepta planiranja (proof of concept).
Uvajanje	Analizira uporabniške potrebe. Izdela strategijo za podporo zmogljivostim. Oceni celotno načrtovanje s stališča uporabnosti. Določi čas in obremenitev za izgradnjo sistema za podporo uporabnikom. Izvede testiranje uporabnosti za vse uporabniške vmesnike.
Testiranje	Oceni načrtovanje in testira značilnosti. Pripravi plan in urnik za testiranje značilnosti. Pripravi metode in merila za odkrivanje hroščev. Izdela strategijo testiranja.
Logistično upravljanje	Oceni načrtovanje s stališča namestitve, upravljanja, vzdrževanja in celotnih stroškov vzdrževanja. Izdela urnik in plan namestitve ter vzdrževanja.

[6]

3.2.3. Metode dela

V uporabi je proces načrtovanja MSF (MSF Design proces). Ta predstavlja učinkovito orodje, ki zagotavlja, da arhitektura aplikacije izraža potrebe uporabnikov in poslovnega procesa. Predvsem učinkovit je pri razvoju večnivojskih aplikacij, in je pravzaprav usmerjen k načrtovanju aplikacij takega tipa. Proces načrtovanja MSF je sestavljen iz treh različnih načinov načrtovanja: konceptualno, logično in fizično načrtovanje. Vsak način generira model z enakim imenom: model konceptualnega načrtovanja, model logičnega načrtovanja in model fizičnega načrtovanja. [2]

Tabela 3: Modeli načrtovanja

Načrtovanje	Perspektiva	Akcija
Konceptualno	Vidi problem s perspektive uporabnika in poslovnega procesa.	Definira problem in rešitev kot scenarije.
Logično	Vidi rešitev s perspektive projektnega tima.	Definira rešitev kot množico med seboj povezanih servisov.
Fizično	Vidi rešitev s perspektive razvijalcev.	Definira tehnologijo in servise rešitve.

Proces načrtovanja MSF vodi od otipljivih zahtev (primeri uporabe) do abstraktnega načrta aplikacije (konceptualno načrtovanje), nato do racionaliziranega načrta aplikacije (logično načrtovanje), do konkretnega načrta aplikacije (fizično načrtovanje) in končno do resničnega produkta.

Slika 3: Modeli načrtovanja

Tri vrste načrtovanja imajo naslednje lastnosti:

- prekrivanje (vsa tri načrtovanja lahko potekajo paralelno),
- iterativnost (vsako načrtovanje ima svoj cikel: preverjanje, testiranje načrtovanja in ponovno načrtovanje),
- spiralnost (vsaka posamezna iteracija vseh treh načrtovanj predstavlja napredek v smeri mejnika *potrditev projektnega plana*).

V vsakem trenutku morajo biti vsi trije načrti med seboj skladni. Če je bila izvedena sprememba na fizičnem načrtu, je potrebno ustrezno spremeniti tudi logični in konceptualni načrt in obratno. [2]

3.2.4. Konceptualno načrtovanje

Cilj: Identifikacija poslovnih potreb in razumevanje, kakšno je delo uporabnikov in kaj pri tem potrebujejo.
Izhod: Množica modelov z informacijami, primeri uporabe, scenariji primerov uporabe in dokumenti, ki predstavljajo trenutno in bodoče stanje.
Konceptualno načrtovanje generira scenarije, ki izražajo celotne in točne zahteve z vključenimi strankami, uporabniki in drugimi vpletenimi in predstavlja jezik uporabnikov. Začne se lahko že v *fazi vizije* in sicer takrat, ko so člani tima enotni v izjavah vizije. Konceptualno načrtovanje ne vsebuje pristopa oziroma tehnologije, potrebne za izgradnjo rešitve. Omejuje se samo na grobe skice in scenarije za izgradnjo rešitve. To so enostavno razumljivi modeli, pripravljeni s strani stranke, uporabnikov in načrtovalcev.
Konceptualno načrtovanje sestoji iz treh korakov: raziskava, analiza in racionalizacija. [1]

3.2.4.1. Raziskava

Pred začetkom procesa konceptualnega načrtovanja mora tim najprej določiti cilj njihovih raziskav. Tako naj se tim najprej omeji na opis glavnih procesov ustreznega poslovnega področja. To naj vključuje:

- opis bistvenih poslovnih procesov in njihovih robnih pogojev skupaj s funkcionalnimi elementi poslovanja,
- grob opis poslovnih transakcij znotraj teh poslovnih procesov in
- opis stranke in uporabnikov.

Za opis in prikaz teh aktivnosti se pri uporabi notacij UML uporabljajo diagrami uporabe, diagrami aktivnosti in diagrami sodelovanja.
Bistven poslovni proces:

- vodi poslovanje,
- direktno naslavlja strateške usmeritve in tekmovalnost,
- ima lastnike in stranke, ki jih lahko identificiramo,
- je definiran tako, da je razumljiv zunanjim strankam ali dobaviteljem kakor tudi osebju znotraj podjetja,
- je diskretno oziroma minimalno odvisen od ostalih bistvenih procesov

Bistvo raziskave je v pridobitvi celotne in točne slike, kjer je kvaliteta pridobljenih informacij pomembnejša od kvantitete pridobljenih podatkov.
Raziskavo je potrebno narediti tudi na uporabnikih oziroma skupini uporabnikov. Identificirati je potrebno čim več skupin uporabnikov, vključujoč lastnike organizacije, uporabnike znotraj organizacije in uporabnike zunaj organizacije, kot so dobavitelji in stranke. Za vsako skupino je potrebno izdelati svoj profil, ki pove, kakšna je njena vloga v organizaciji, njen oddelek, lokacija in njeno sodelovanje pri specifičnih aktivnostih. Te

skupine in uporabnike pa je potrebno tudi povezati s predhodno zbranimi procesi in aktivnostmi. [2]

3.2.4.2. Analiza

Prva naloga drugega koraka, koraka analize, je potrditev rezultatov koraka raziskave, najpogosteje s skupinskim usklajevanjem. Ko je raziskava usklajena, je naslednja naloga priprava modelov, ki ponazarjajo kontekst, procesni tok in vrstni red nalog. Obstajata dva modela: diagrami primerov uporabe s scenariji primerov uporabe in diagrami aktivnosti. [2]

3.2.4.3. Racionalizacija

V tem koraku je bistvenega pomena določitev izboljšav, kjer je to možno. Kjer je mogoče, mora celoten tim, po možnosti tudi z zunanjo pomočjo, optimizirati poslovne procese. Kaj naj se optimizira? Cilj je v eliminaciji:

- neefektivnosti,
- neopisanih in nepotrebnih korakov,
- odvečnih in neefektivnih praks in procesov,
- nefunkcionalne politike in
- prevoza in zamudnega časa.

Tim si mora predstavljati in opisati bodoče stanje. Ko je bodoče stanje vizualizirano, se lahko na podlagi tega pripravi ustrezne nove scenarije.
Pri optimizaciji procesov je potrebno upoštevati naslednje osnove načrtovanja:

- kršitev pravil,
- upoštevanje ciljev učinkovitosti delovanja,
- načrtovanje na podlagi produktov in storitev,
- odstranitev birokratskih in drugih ovir,
- izboljšava produktivnosti,
- kje lahko tehnologija nudi podporo in
- drobljenje procesov na manjše procese.

Po izdelavi novih scenarijev je zadnji korak potrditev le teh, torej zagotovitev, da rešujejo poslovni problem. Tim doseže to z:

- izdelavo sistema preverjanja ustreznosti koncepta različice sistema,
- uporabo sistema preverjanja ustreznosti koncepta različice za predstavitev načrta uporabniškega vmesnika,
- pridobitvijo povratnih informacij o uporabnosti in procesu in s ponavljanjem, dokler stranka in uporabniki niso zadovoljni.

Sistem sistema preverjanja ustreznosti koncepta naj ima v zgodnjem načrtovanju samo osnovne značilnosti, načrt uporabniškega vmesnika in okvirno strukturo sistema. Zavzame lahko naslednje oblike:

- aplikacija, ki prikazujejo osnovno funkcionalnost,
- snemalne knjige (papirnate ali ekranske oblike),
- papirnate prototipe celotne strukture in uporabniške interakcije,
- MS PowerPoint dokument, ki prikazuje osnovne elemente sistema in demonstracijo navigacije skozi sistem za eno ali več opravil.

Ob napredovanju načrtovanja lahko prototipi dobijo definirano podobo, ki omogoča timu ocenitev vizualne oblike in nekaterih drugih načrtovalskih podrobnosti, predvsem uporabniškega vmesnika.
Pomembno je, da se ne poskuša preoblikovati celotnega poslovnega procesa z začetno različico aplikacije. To se lahko doseže v kasnejših scenarijih, ki nudijo dodatne izboljšave modeliranega poslovnega procesa. [2]

3.2.5. Logično načrtovanje

Cilj: Logično načrtovanje je proces opisovanja rešitve v organizacijskem, strukturnem in sintaktičnem smislu in sodelovanju njenih delov s perspektive projektnega tima. Predstavlja pogled na rešitev, kot jo vidi projektni tim.
Izhod: Izhodišče je množica poslovnih objektov z ustreznimi storitvami, atributi in povezavami, dizajn visokonivojskega uporabniškega vmesnika in dizajn logičnega podatkovnega modela.
Logično načrtovanje upravlja in odstranjuje kompleksnost z definiranjem rešitve, opisovanjem njenih delov in opisov, kako ti deli sodelujejo med seboj in tako rešijo problem. Odkriva in odstranjuje nekonsistenco konceptualnega modela. Odstranjuje redundantnost in identificira potencialno ponovno uporabo. Predstavlja tudi temelj za fizično načrtovanje. Razločno predstavi pogled na rešitev celotnemu timu.
Logično načrtovanje je neodvisno od fizične implementacije. Primarno se je potrebno osredotočiti na to, kaj naj sistem dela.
Logično načrtovanje vsebuje dva koraka: analiza in realizacija. [1]

3.2.5.1. Analiza

Cilj analize je pretvorba scenarijev, izdelanih v konceptualnem načrtovanju, za uporabo v logičnem načrtovanju. Moduli so bistveni primeri uporabe sistema, kakor tudi servisi ali akcije, ki se v njih izvajajo in njihove povezave. Modul je logična enota, ki abstraktno predstavlja primere uporabe in njihove scenarije. Tim mora za vsak modul identificirati njegove servise, objekte, atribute in povezave. [2]

3.2.5.2. Racionalizacija

Prva naloga racionalizacije je kreiranje do sedaj še ne kreiranih servisov in objektov. Kreiranje pa je potrebno bodisi zaradi predvidene potrebe ali zaradi potrebe po kontroli. [2]

3.2.6. Fizično načrtovanje

Cilj: Uporabiti želimo tehnologijo na že pripravljenem logičnem načrtu upoštevajoč elemente implementacije in učinkovitosti delovanja.
Izhod: Množica komponent, načrt uporabniškega vmesnika za določeno platformo in fizični podatkovni načrt.
Fizični načrt predstavlja osnovo funkcionalne specifikacije, ki jo *razvoj, testiranje in logistično upravljanje* uporabijo za zagotavljanje kvalitete. To je pomembno, kajti fizični načrt predstavlja zadnjo možnost preverjanja pred kodiranjem. Razvoj in dostava se lahko pričneta tudi že pred zamrznitvijo fizičnega načrta, kar omogoča naknadno čiščenje načrta, vendar pa mora biti zamrznjen pred stabilizacijo rešitve.
Fizični načrt ponuja osnovo drugim nalogam *faze planiranja*:

- izdelava približkov za ceno, terminski plan in plan virov,
- osveževanje analize tveganja.

Fizično načrtovanje sestoji iz štirih korakov: raziskava, analiza, racionalizacija in implementacija. [1]

3.2.6.1. Raziskava

Cilja koraka raziskave sta dvosmerna in sicer:

- določitev infrastrukturnih omejitev in fizičnih zahtev aplikacije in
- upravljanje s tveganjem konflikta med fizičnimi infrastrukturnimi omejitvami in fizičnimi zahtevami aplikacije.

Celoten tim pripravi spisek zahtev in ustreznih omejitev. Seznam je potrebno pripraviti z različnih perspektiv kot so:

- učinkovitosti delovanja,
- razmerje med ceno in dobičkom,
- dostava,
- vzdrževanje,
- izbira tehnologij,
- zanesljivost,
- dostopnost in
- varnost.

Raziskava trenutne fizične infrastrukture poteka z uporabo topologij, predstavljenih kot načrti, ki predstavljajo različne aspekte infrastrukture. Za potrebe planiranja je smiselna priprava omrežnih, podatkovnih in komponentnih topologij, ki predstavljajo trenutno stanje infrastrukture. [2]

3.2.6.2. Analiza

Poglavitna naloga koraka analize je izbira kandidatov tehnologije za implementacijo na podlagi razumevanja zahtev aplikacije. To so torej kandidati, odločitev je izvedena kasneje. Ko so potencialne tehnologije izbrane, lahko tim pripravi predhodni model dostave.
Pri ocenjevanju tehnologij naj tim na prvem mestu upošteva poslovne zahteve, šele nato zahteve arhitekture podjetja ter na koncu še tehnološke zahteve.

Glede na poslovne zahteve je potrebno upoštevati:

- sposobnost, ali bo tehnologija sploh zadovoljevala poslovne zahteve,
- ceno produkta (potrebno se je zavedati celotne cene: cene razvijalcev, strežnikov, licenc, nadgradenj),
- izkušnje (potrebno se je zavedati, da imajo lahko izkušnje ali pa njihovo pomanjkanje velik vpliv, katere izkušnje so na voljo v obliki tečajev),
- zrelost ali inovacijo,
- vzdrževanje (podpora je potrebna tako tehnologiji, kakor tudi rešitvi, izdelani s pomočjo te tehnologije, možnosti podpore so prodajalec, zunanja oskrba in informacijska deska).

V razmislek pridejo še dostava, konkurenčna prednost, čas trženja in zaznavanje industrije.

Glede na arhitekturo podjetja je potrebno upoštevati:

- **usklajenost z arhitekturnimi cilji podjetja**
  Aplikacija naj ustreza zastavljenim arhitekturnim ciljem podjetja. Cilji bazirajo na štirih principih in modelih arhitekture podjetja: poslovanje, aplikacija, informacija in tehnologija.
- **vdanost arhitekturi podjetja**

Arhitektura podjetja bo določala tako plane trenutnega kakor tudi bodočega stanja.

- **možnost razširjanja**
  Skalabilnost aplikacije je lahko upoštevana znotraj širjenja poslovanja.
- **večuporabnost**
  Aplikacija mora sodelovati tudi z drugimi aplikacijami znotraj organizacije.

Glede na tehnologijo je potrebno upoštevati:

- **programski jezik**
  Pri izbiri programskega jezika za razvoj komponent je potrebno za različne naloge razmisliti o različnih jezikih.
- **standarde za komunikacijo med komponentami**
  Pri izbiri standardov za komunikacijo med komponentami je potrebno upoštevati integracijo med platformami z vidika zmogljivosti in učinkovitosti delovanja.
- **metode za dostop do podatkov**
  Upoštevati je potrebno predvsem učinkovitosti delovanja, standarde in bodoče usmeritve kakor tudi raznolikost podatkovnih baz in upravljanja z njimi.
- **sistemske storitve**
- **operacijski sistem**
  Pri izbiri operacijskega sistema lahko storitve, ki jih le ta ponuja, znatno znižajo potrebe po kodiranju aplikacije. Poleg tega lahko le ta poseduje ustrezne varnostne mehanizme za skalabilnost.

Misleč na kandidate za tehnologijo lahko sedaj tim izdela predhodni model dostave, ki je sestavljen iz omrežne, podatkovne in komponente topologije. Na tej točki so to šele predlagane in ne izbrane topologije.

Omrežna topologije predstavlja infrastrukturni načrt, ki ponazarja lokacije strojne opreme in medsebojne povezave.

Podatkovna topologija je predstavljena z načrtom distribucije podatkov, ki ponazarja lokacije podatkovnih baz v povezavi z omrežno topologijo.

Komponenta topologija je predstavljena z načrtom distribucije komponent, ki ponazarja lokacije komponent in njihovih servisov v povezavi z omrežno topologijo.

Vsi načrti ponazarjajo trenutno stanje in so bili razviti že v prejšnjem koraku analize, tu pa so mogoče potrebne izboljšave, ki podpirajo novi fizični načrt. [1]

3.2.6.3. Racionalizacija

Raziskava in analiza sta končani in čas je za dokončanje fizičnega načrtovanja in sicer določitev pakiranja in distribucije.

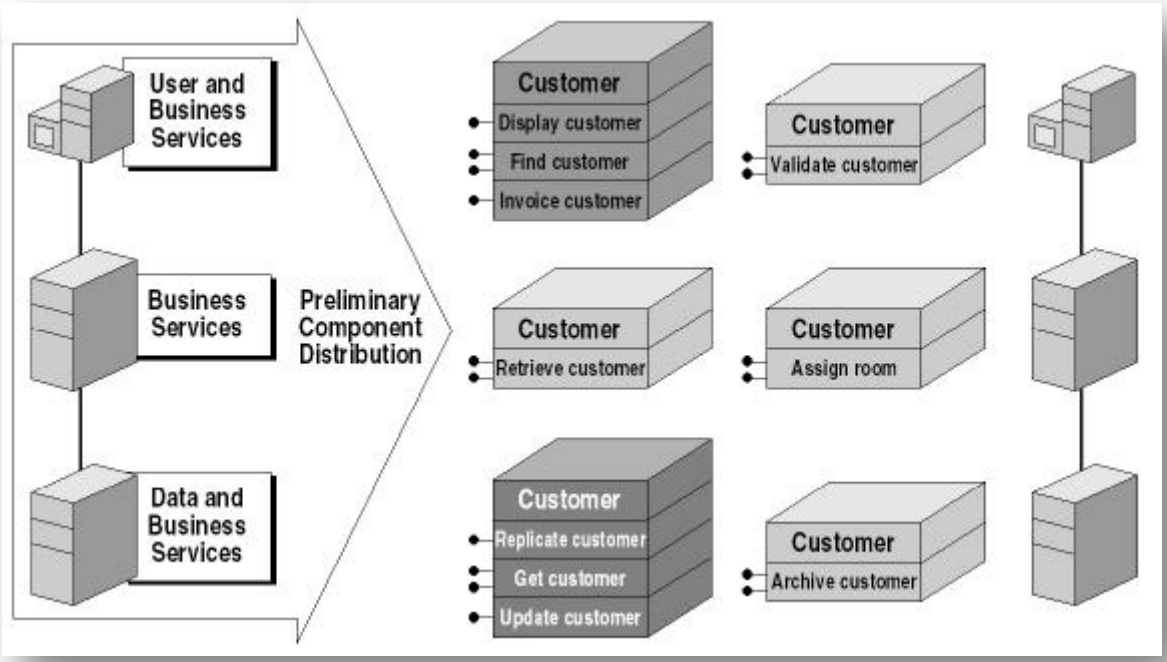
Pri določanju strategije pakiranja in distribucije naj tim sledi naslednjim enostavnim korakom. Najprej je potrebno razmisliti o različnih načelih pakiranja ali razlogih za izbor določene strategije. Med te spadajo naslednji:

- kategorija storitve,
- skalabilnost,
- učinkovitost delovanja,
- upravljivost,
- ponovna uporaba,

- poslovni kontekst in
- razdrobljenost.

Drugi korak je v uskladitvi strategije s programskim modelom.
Tretji korak je določitev načrtovalskih izmenjav, ki vplivajo na strategijo.

Tim je sedaj pripravljen na začetek definiranja komponent. Čeprav za to obstaja več načinov, je priporočljivo, da je osnovna ideja izdelana na podlagi aplikacijskega modela MSF. Ta pa predstavlja tri nivoje storitev: uporabniški, poslovni in podatkovni nivo. Storitve so porazdeljeno glede na topologijo, kakor kaže slika. Tim lahko nato pakira kandidatke za komponente iz iste logične storitve na isto mesto v topologiji.



Slika 4: Trije nivoji storitev

Ko je komponenti model zaključen, lahko tim zaključi tudi porazdelitev teh komponent po nivojih. Ni pa nujno, da uporaba treh logičnih nivojev ne pomeni porazdelitev le teh na tri fizične lokacije. Povsem možna je rešitev, kjer N-nivojska aplikacija teče na enem računalniku. Prav tako pa lahko distribucijska strategija za kompleksno aplikacijo pomeni porazdelitev na deset, dvajset ali več fizičnih lokacij.

Med racionalizacijo naj tim preverja komponente in njihovo pakiranje ter dostavo. To naj izvršita predvsem vlogi: *razvoj* in *testiranje*. Preverjajo naj, ali:

- načrtovane komponente vsebujejo vse storitve načrtovanja v logičnem načrtu,
- so bile upoštevane vse odvisnosti komponent/storitev,
- porazdelitev komponent glede na topologijo ustreza načrtovani,
- so storitve grupirane po komponentah tako, da so v ravnovesju z fizično lokacijo, pakiranjem in omejitvami tehnologije,
- komponentni plan upošteva razširljivost (navzgor kakor tudi navzdol). [2]

3.2.6.4. Implementacija

Zadnji korak fizičnega načrtovanja je implementacija. V tem koraku tim specificira programski model, ki ga nato *razvoj* uporabi pri razvoju. Tu so specificirani tudi vsi vmesniki komponent in njihova interna struktura.

Programski model, ki mu lahko rečemo tudi programska specifikacija ali standard, zagotavlja naslednje:

- predpisuje, kako uporabljati izbrane tehnologije,
- določa dizajnerske smernice pri specifikaciji komponent, kar pripomore k večji konsistenci celotnega projekta,
- uporablja različne pristope k reševanju različnih aspektov rešitve.

Pri izdelavi programskega modela je potrebno upoštevati več aspektov kot so:

- **tehnologija implementacije**
  Programski jezik, API, strežniki in strežniške tehnologije, …
- **objekti z ali brez stanja**
  Objekt s stanjem hrani stanje, pridobljeno s strani več odjemalcev, medtem ko objekt brez stanje ne hrani ničesar.
- **klici funkcij v-proces proti iz-procesa**
  Ali naj se funkcija izvaja v svojem ali v kličočem procesu?
- **povezani proti nepovezanim modelom (sinhronski proti asinhronskim programskim modelom)**
  Sinhronski programski model blokira nadaljevanje izvajanja kličoče komponente, dokler klicani vmesnik ne zaključi z delom. Asinhronski model pa dopušča komponentam pošiljanje sporočil drugim komponentam in nadaljevanje izvajanja.
- **nitni model**
  Izbira nitnega modela je zelo težka, ker je odvisna od funkcije, ki jo komponenta izvaja.
- **upravljanje napak**
- **varnost**
  Vsaka komponenta ima štiri možnosti zaščite: na podlagi komponente bodisi na nivoju metode, nivoju vmesnika ali pa na nivoju komponente; na podlagi podatkovne baze; na podlagi uporabnika bodisi sistemsko ali pa programsko realizirano; na podlagi vlog.
- **dostava**
  Pri uporabi treh logičnih nivojev ne pomeni tudi treh fizičnih. Logični nivoji bodo razporejeni po fizičnih nivojih.

Ko je programski model začrtan, je tim pripravljen na specifikacijo zunanje strukture komponent. Ta pa je predstavljena s pomočjo vmesnikov:

- je dogovor, ki predstavlja povezavo ponudnik-odjemalec med komponentami,
- je način za dostop do vsebovanih storitev,
- predstavlja eno ali več storitev,
- vsebuje atribute vsebovanih objektov.

Specifikacija vmesnika komponente mora vsebovati vse možne načine dostopa do komponente, po možnosti s pripadajočimi primeri za vsak način.

Pri izdelavi vmesnikov komponent je potrebno upoštevati naslednja dejstva:

- izdelan vmesnik predstavlja dogovor, ki naj bi bil trajen,
- sprememba obstoječega vmesnika naj bi bila izdelana kot nova komponenta ali nov vmesnik,
- podatkovni tipi izdelanih atributov morajo biti podprti s strani storitvenega vmesnika odjemalca.

Vmesnik predstavlja dogovor, ki določa parametre, podatkovne tipe, sodelovalne standarde in opise samega vmesnika. Stopnja specificiranosti je odvisna od uporabniških zahtev in končno od zahtev ponovne uporabe.

Končno je tim pripravljen za specificiranje notranje strukture komponent. Pri definiranju notranje strukture komponente je potrebno upoštevati več faktorjev. Najpomembnejši je gotovo programski jezik ali orodje za implementacijo komponente.
Ko je notranja struktura komponent specificirana, so osnove za implementacijo vzpostavljene. [2]

3.2.7. Mejnik odobritev projektnega plana in izdelki faze planiranja

Cilj faze *planiranja* je doseg mejnika *odobritev projektnega plana*, ki predstavlja glavni cilj tega koraka in dogovor oziroma pogodbo med stranko, naročnikom in timom.
Za doseg mejnika *odobritev vizije* so potrebni naslednji izdelki:

- funkcionalna specifikacija,
- glavni projektni plan in urnik in
- revidiran dokument tveganj.

Na tem mejniku se tim še vedno lahko odloči za opustitev ali nadaljevanje projekta. Kajti dobro zaključena *faza planiranja* lahko timu in stranki oziroma naročniku zagotovi nadaljevanje projekta z zaupanjem in prepričanjem, da je bilo storjeno vse za uspešnost projekta. [2]

3.2.7.1. Funkcionalna specifikacija

Funkcionalna specifikacija je glavni izdelek *faze planiranja*. Predstavlja podrobno zbirko specifikacij za projekt in služi kot pogodba med stranko in projektnim timom. Pretirano je reči, da se vse dosedanje delo manifestira v funkcionalni specifikaciji in da bo nadaljnje delo izhajalo iz nje.
Lastnik funkcionalne specifikacije je programsko upravljanje, ki je tudi zadolženo za dokončanje le te. To pa ne pomeni, da mora programsko upravljanje napisati celotno funkcionalno specifikacijo, ampak mora odsevati razumevanje produkta celotnega tima.

Vsebina funkcionalne specifikacije

Funkcionalna specifikacija naj vsebuje vse spodaj naštete elemente:

Tabela 4: Funkcionalne specifikacije

Element	Opis
Povzetek vizije	Kaj tim želi, da bi produkt bil, njen zagovor in ključne omejitve.
Cilji načrtovanja	Kaj želi tim s projektom doseči. Razvoj uporabi te cilje kot izhodišče pri odločitvah o zmogljivosti, stabilnosti, uporabnosti in dostopnosti.
Zahteve	Kaj stranka, uporabniki in naročnik mislijo, da bo projekt. Te zahteve naj bodo razvrščene po prioritetah.
Povzetek uporabljivosti	Kdaj bo produkt uporabljen in kdo ga bo uporabljal.
Značilnosti	Kaj točno je produkt dela. Značilnosti produkta, razvrščene po prioritetah.

Odvisnosti	Od česa je produkt odvisen. Spisek zunanjih objektov, od katerih bo produkt odvisen.
Povzetek urnika	Opis urnika. Povzetek projektnega urnika z poudarkom na vmesne mejnike, vmesne produkte in datum zaključka.
Tveganja	Kakšna so tveganja.
Dodatki	Vse ostalo. Množica izdelkov procesa načrtovanja, ki jih je tim uporabil pri pripravi funkcionalne specifikacije.

Pregled in sprejetje funkcionalne specifikacije

Glavni cilj revizije je pridobiti mnenja vseh vlog. Končni cilj priprave funkcionalne specifikacije pa je njeno sprejetje oziroma konsenz vseh vlog. Projektni tim, stranka oziroma naročnik naj potrdijo, da funkcionalna specifikacija točno in pravilno dokumentira pričakovanja projekta. [2]

Strinjanja vlog pri funkcionalni specifikaciji:

Tabela 5: Vloge in njihove zadolžitve

Vloga	Zadolžitve
Stranka	Produkt zadostuje poslovnim potrebam. Stranka je pripravljena sprejeti plan in vire za razvoj za obseg, predstavljen v funkcionalni specifikaciji.
Produktno upravljanje	Produkt zadostuje znane zahteve. Produktno upravljanje verjame, da funkcionalna specifikacija odseva produkt, ki bo zadostoval znanim zahtevam, ko bo izdelan.
Projektno upravljanje	Zadolžitve in plani tima so realistični. Projektno upravljanje verjame, da je jasno, kdo je zadolžen za posamezno funkcijo in da je sprejeti plan realističen.
Razvoj	Produkt je možno razviti. Razvoj je uspešno spremljal tveganja razvoja in verjame, da so tveganja za dosego zastavljenega datuma dokončanja realistična.
Uvajanje	Produkt je uporaben in potrebe po podpori uporabniku so definirane. Uvajanje je seznanjeno, kdo so uporabniki in kako bo produkt uporabljen in podprt.
Testiranje	Produkt je možno testirati in stabilizirati. Testiranje ima definirano strategijo za testiranje vseh aspektov funkcionalne specifikacije.
Logistično upravljanje	Produkt je možno vzdrževati in namestiti. Logistično upravljanje je seznanjeno z organizacijskimi, aplikacijskimi in sistemskimi izhodišči in jamči namestitev produkta.
Vse vloge	Vsi se strinjajo z datumom zaključka.

3.2.7.2. Projektni plan

Glavni projektni plan nam pove, kako bo produkt izdelan z združevanjem detajlnih planov članov projektnega tima. Namen glavnega projektnega plana je:

- omogočanje uskladitev tima in plana dela za določeno vlogo,
- opisovanje, kako bodo tim in vloge izvrševale svoje vloge,
- omogočanje sinhronizacije planov celotnega tima.

Lastnik glavnega projektnega plana je projektno upravljanje, ker predstavlja glavnega koordinatorja planiranja in dejanskega dela na projektu. Vseeno pa je vsaka vloga zadolžena za izdelavo in vzdrževanje lastnega realnega projektnega plana glede na glavni plan. [2]

Glavni projektni plan naj vsebuje naslednje spodaj naštete elemente:

Tabela 6: Vsebina projektnega plana

Element	Opis
Razvoj	Opisuje, kako bo razvoj izdelal vse, kar je zapisano v funkcionalni specifikaciji. Opisuje stvari kot so orodja, metodologije, primeri uporabe, zaporedja dogodkov in metode za testiranje.
Testiranje	Vsebuje strategijo testiranja: specifična področja, ki morajo biti testirana ter ustrezne vire (oprema, ljudje).
Izobraževanje	Vsebuje strategijo in plan za izdelavo vsega potrebnega testnega materiala.
Podpora uporabnikom	Vsebuje strategijo in podrobnosti za razvoj vsega, kar bodo uporabniki potrebovali za lažje delo (čarovniki, priročnik za uporabo, …).
Komunikacija	Vsebuje marketinško strategijo in aktivnosti promocije uporabnikom.
Namestitev	Vsebuje strategijo in podroben plan za pripravo končnih uporabnikov in administratorjev pred in med namestitvijo.

[2]

3.2.7.3. Revidirani dokument tveganj

Ob koncu faze *planiranja* mora tim imeti precej jasno predstavo o tveganjih, povezanih s projektom ter njihovimi posledicami ter njihovo prioriteto.
Projektno upravljanje si dokument lasti in je hkrati odgovorno da je revizija končana in točna.
Revidirani dokument tveganj vsebuje zbirko vseh tveganj različnih članov tima.
Dokument je v pomoč pri usklajevanju seznama tveganj celotnega tima. [1]

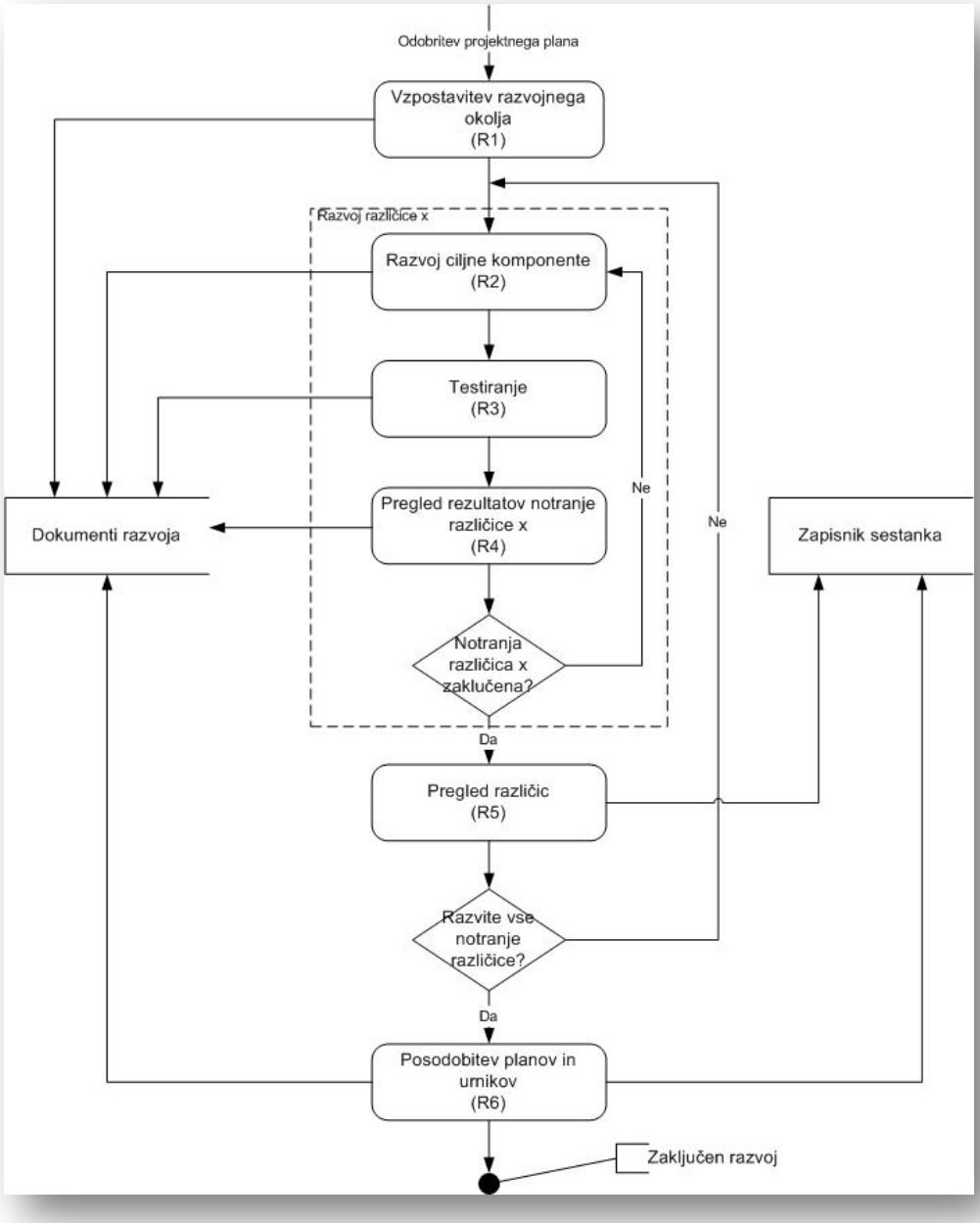
3.3. Razvoj

3.3.1. Splošno

V *fazi razvoja* si zastavimo vprašanje »Kako uspešno izvesti razvojni proces, da dobimo pravi produkt ob pravem času?«. Odgovor najdemo v skupnemu razumevanju vizije produkta. Da bi bila *faza razvoja* še bolj uspešna, mora tim poleg razumevanja vizije upoštevati tudi realne omejitve razvoja delujočega produkta. Po zaključku faze razvoja in dosegu mejnika imamo zaključen razvoj z naslednjimi lastnostmi:

- implementirane so vse značilnosti produkta, čeprav ne v celoti optimizirane,
- izvedeno je bilo osnovno testiranje in pripravljen je spisek tekočih hroščev (ni nujno, da so že odpravljeni),
- člani tima in naročnik se strinjajo, da vključene značilnosti ustrezajo viziji in planiranju in so uspešno implementirane,
- za testiranje in stabilizacijo so pripravljeni okvirni materiali uporabniških učinkovitosti delovanja.

Pri prehodu iz planiranja v razvoj tim ne more razviti vseh funkcionalnosti hkrati, temveč mora biti kodiranje razdeljeno v več različic (segmentov). Le te je potrebno ustrezno upravljati in nadzirati in na koncu združiti v končni produkt.
Razvoj predstavlja približno 35 do 40% celotnega projekta. [2]



Slika 5: Shema razvoja

3.3.2. Kdo ga izvede?

Spodnja tabela opisuje tipične zadolžitve vlog v *fazi razvoja*. Vodja vsake vloge skrbi za izvršitev zadanih nalog in komunikacijo z ostalimi člani tima.

Tabela 7: Tipične zadolžitve v fazi razvoja

Vloga	Zadolžitve
Produktno upravljanje	Zadolženo je za upravljanje z zahtevami in pričakovanji stranke. Prav tako skrbijo za informiranje stranke in zunanjih naročnikov. Zadolženo pa je tudi za pripravo na alfa in beta različico.
Projektno upravljanje	Vodi komunikacijo preko celotnega projektnega tima in koordinira interne različice prek celotnega tima. Prav tako je zadolženo za možno revizijo dokumentov, pripravljenih v *fazi planiranja*.
Razvoj	Izdela vso kodo, potrebno za implementacijo vseh značilnosti produkta. Prav tako je zadolženo za osnovno testiranje kode in značilnosti produkta.

Uvajanje	Izvede osnovno testiranje uporabnosti produkta in prične s testiranjem uporabniških učinkovitosti delovanja. Koordinira uporabniški forum za alfa in beta različici produkta. Skrbi za izdelavo uporabniških priročnikov in dokumentacije za vzdrževanje.
Testiranje	Izdela podroben plan testiranja (scenariji, podatki, skripti,…) za izvedbo osnovnega funkcionalnega testiranja. Le to je izvedeno v *fazi razvoja* za potrditev internih različic. Osnovna naloga testiranja je identificiranje in sledenje hroščem ter vodenje celotne dokumentacije testiranja.
Logistično upravljanje	Nudi logistično podporo pri razvoju. Pripravi tudi ves material in dokumentacijo, potrebno za namestitev in osnovno vzdrževanje alfa in beta različic produkta.

[6]

3.3.3. Metode dela

Proces razvoja MSF je sestavljen iz treh korakov: *analiza* in *racionalizacija*, *implementacija* in *validacija*. [2]

3.3.3.1. Analiza in Racionalizacija

V tej *fazi razvoja* ima tim konkreten plan akcij z nekaj spremenljivkami. V tej fazi sicer dodatne analize niso potrebne, a naj se faza kljub temu prične z analizo trenutnega načrta aplikacije.

Analizirati je potrebno projektni plan in urnik ter identificirati vire, ki jih je potrebno dodeliti za kodiranje. Po pregledu skupkov značilnosti, pripravljenih v *fazi planiranja*, razvojni tim razdeli razvoj posameznih značilnosti članom. Tu je potrebno upoštevati tudi delitev aplikacije v servisne plasti.

Tim za testiranje analizira načrt produkta in določi, kdo bo izvedel posamezen test in pregleda primere uporabe, da zagotovi ustrezno testiranje uporabnosti produkta. Pripravi tudi izčrpno zbirko testnih scenarijev na podlagi primerov uporabe, fizičnega načrta in nefunkcionalnih zahtev, kot so aplikacijske in uporabniške zahteve po učinkovitosti delovanja.

Po izdelavi posamezne interne različice naj tim analizira odziv uporabnikov in tima za testiranje in vzdrževanje, s čemer se lahko določi uspešnost trenutne interne različice produkta. Vsi hrošči in drugi problemi produkta morajo biti najprej znani in nato ustrezno odpravljeni. [2]

3.3.4. Implementacija

Glavna naloga je seveda implementacija izvorne kode produkta, ki jo pripravi razvojni tim. Poleg tega pa mora uvajalni tim pripraviti dokumente, potrebne za vzdrževanje aplikacije kakor tudi uporabnikov. Poleg tradicionalnih uporabniških in namestitvenih dokumentov se lahko tim osredotoči tudi na on-line pomoč ter razne vodiče ter programske čarovnike za izboljšavo uporabnosti in podpore. [2]

3.3.4.1. Validacija

Validacija je delo celotnega tima. Posebno pa je to odgovornost razvojnega tima in tima za testiranje. Validacija poteka na področjih uporabnosti, preformans, sledenja hroščem in ideji o produktu brez napake.

Zaradi več internih različic v *fazi razvoja* je potrebno v validacijo vključiti kar precejšen del tima. Za večjo učinkovitost pri validaciji kode naj razvojni tim in tim za testiranje v čim večji meri avtomatizirata proces testiranja. [2]

3.3.5. Mejnik zaključen razvoj ter izdelki faze razvoja

Glavni cilj *faze razvoja* je doseg mejnika *zaključen razvoj*. Za doseg mejnika *zaključen razvoj* so potrebni naslednji izdelki:

- **revidirana funkcionalna specifikacija**
  Prikazuje dejansko stanje izdelane aplikacije z vsemi spremembami v *fazi razvoja*.
- **revidirani projektni plan in urnik**
  Osvežen plan in urnik, ki predstavlja dejanski plan in urnik implementacije.
- **revidirani dokument tveganj**
  Dodana so še podrobna tveganja posredovana od posameznih članov tima. [2]

3.3.5.1. Interne različice produkta

Interne različice sledijo celotnemu konceptu razvoja v različicah. Obe vključujeta inkrementalno dodajanje funkcionalnosti na že zastavljeno osnovo. Interne različice omogočajo timu stabilizacijo produkta, dosego ciljev kvalitete in vajo za oddajo produkta. Pri uporabi internih različic v *fazi razvoja* je pomembno, da se določi znana osnova, ki se kasneje uporabi kot primerjalno orodje.
Interne različice nastajajo skozi celotno *fazo razvoja*. Vsaka različica je načrtovana tako, da razširja oziroma dopolnjuje prejšnjo, kakor kaže spodnja slika. Neodvisnost različic nudi pogled na različico kakor že na končen produkt, kar še dodatno spodbudi tim v *fazi razvoja*. Revizija interne različice je pomemben del pri usmerjanju tima v učečo se organizacijo, ki uporablja najboljša praksa in se uči iz lastnih napak.

Slika 6: Različice produkta

Vsaki interni različici določimo nivo kakovosti, ki ga mora zadovoljiti, da je dosežen interni mejnik. Nivo kakovosti predstavlja merilno orodje, ki ga tim jasno določi. Vsaka interna različica ima v manjši meri tudi *fazo stabilizacije*, s pomočjo katere je dosežena določena kvaliteta.
Pri razvoju produkta tim s pomočjo zaporedoma sledečih internih različic inkrementalno dodaja skupke značilnosti, vse dokler produkt ni končan. Uspešnost delitve produkta v interne različice oziroma skupke značilnosti je odvisna predvsem od pravilnega planiranja v predhodni *fazi planiranja*. [2]

3.3.5.2. Izvorna koda in izvršilne datoteke

Izvorna koda predstavlja osnovo za končni produkt in je podvržena nadzoru kvalitete. Kvaliteti kode se ne smemo odreči, pa čeprav smo pod pritiskom časa. Upravljanje je zadolženo za izbiro kompromisa med kvaliteto kode in dosego zastavljenega roka. Da bi

zmanjšali možnosti žrtvovanja kvalitete v želji po prihranku časa, mora tim postaviti standarde, ki:

- silijo razvijalce, da uporabljajo metodološki in disciplinirani pristop k kodiranju, ne glede na željo po uporabi bližnjic,
- konstantno opominjajo razvijalce, da je interna kvaliteta kode pomembna,

Standardi kodiranja vsebujejo naslednje elemente:

- poimenovanja,
- obliko (poravnave, zamiki, …),
- komentarje in
- kako kodirati in kako ne. [2]

3.3.5.3. Dokumenti podpore

Dokumenti podpore obsegajo vse od čarovnikov znotraj produkta pa do uporabniške dokumentacije in navodil ter materiala za tečaj. Kakor je zapisal Steve Maguire v knjigi Debugging the Development Process (Microsoft Press, 1994):

> »*Programerji morajo programirati z mislijo na uporabnika. Če programer misli, da je določena naloga neintuitivna ali počasna, je velika verjetnost, da bo enakega mnenja tudi uporabnik.***«** [2]

3.3.5.4. Elementi testiranja

Združujejo različna orodja, ki jih lahko uporabimo pri načrtovanju in razvoju solidnega produkta.

Integrirano testiranje ob mejniku

Postopek testiranja ni omejen samo na *fazo stabilizacije*, saj je bistveni del *faze razvoja*. Specifikacija testiranja je zaključena ob mejniku *zaključen razvoj*, ko je število značilnosti fiksno. Ob koncu *faze razvoja* je produkt v alfa obliki (to nima povezave z alfa in beta testiranjem v *fazi razvoja*). Pri prehodu iz *faze razvoja* v *fazo stabilizacije* prehajamo iz testiranja obsega v testiranje uporabnosti. Primeri testiranja obsega na tipičnem projektu:

- testi celote,
- funkcionalni testi,
- testi prevoda,
- testi nazadovanja (mogoče deluje kaj slabše kakor je prej).

Tipični testi uporabnosti:

- test konfiguracije (produkt dela na ciljni HW in SW opremi),
- test združljivosti (interakcija z drugimi programi),
- test učinkovitosti delovanja in
- test dokumentacije in uporabniške pomoči (pregled napak).

Alfa test predstavlja prvi test celotnega produkta s pomočjo zunanjega vira. Beta testi so testi, izvedeni na demo produktu s pomočjo množice zunanjih virov s ciljem ugotoviti napake, ki jih projektni tim ni našel.

Pri odpravi hroščev je potrebna njihova klasifikacija. Osnovna elementa klasifikacije hroščev sta resnost in prioriteta. Resnost predstavlja vpliv hrošča na celoten produkt, če le ta ni odpravljen. Prioriteta pa je le merilo pomembnosti hrošča na stabilnost produkta, ki ga določi tim. Tipični nivoji klasifikacije resnosti:

- resnost 1: odpoved sistema ali nevarnost izgube podatkov,
- resnost 2: velik problem, hrošč predstavlja resno odstopanje funkcionalnosti,
- resnost 3: manjši problem, hrošč predstavlja manjše odstopanje funkcionalnosti,

- resnost 4: trivialno, v glavnem kozmetični problem.

Tipični nivoji klasifikacije prioritete:
- prioriteta 1: najvišja prioriteta, produkta ni mogoče zaključiti,
- prioriteta 2: visoka prioriteta, produkta ni mogoče zaključiti, vendar tim lahko doseže naslednjo interno različico,
- prioriteta 3: majhna prioriteta, hrošče se odpravi, če je dovolj časa,
- prioriteta 4: najnižja prioriteta, z odpravo teh hroščev se dosežejo izboljšave sistema.

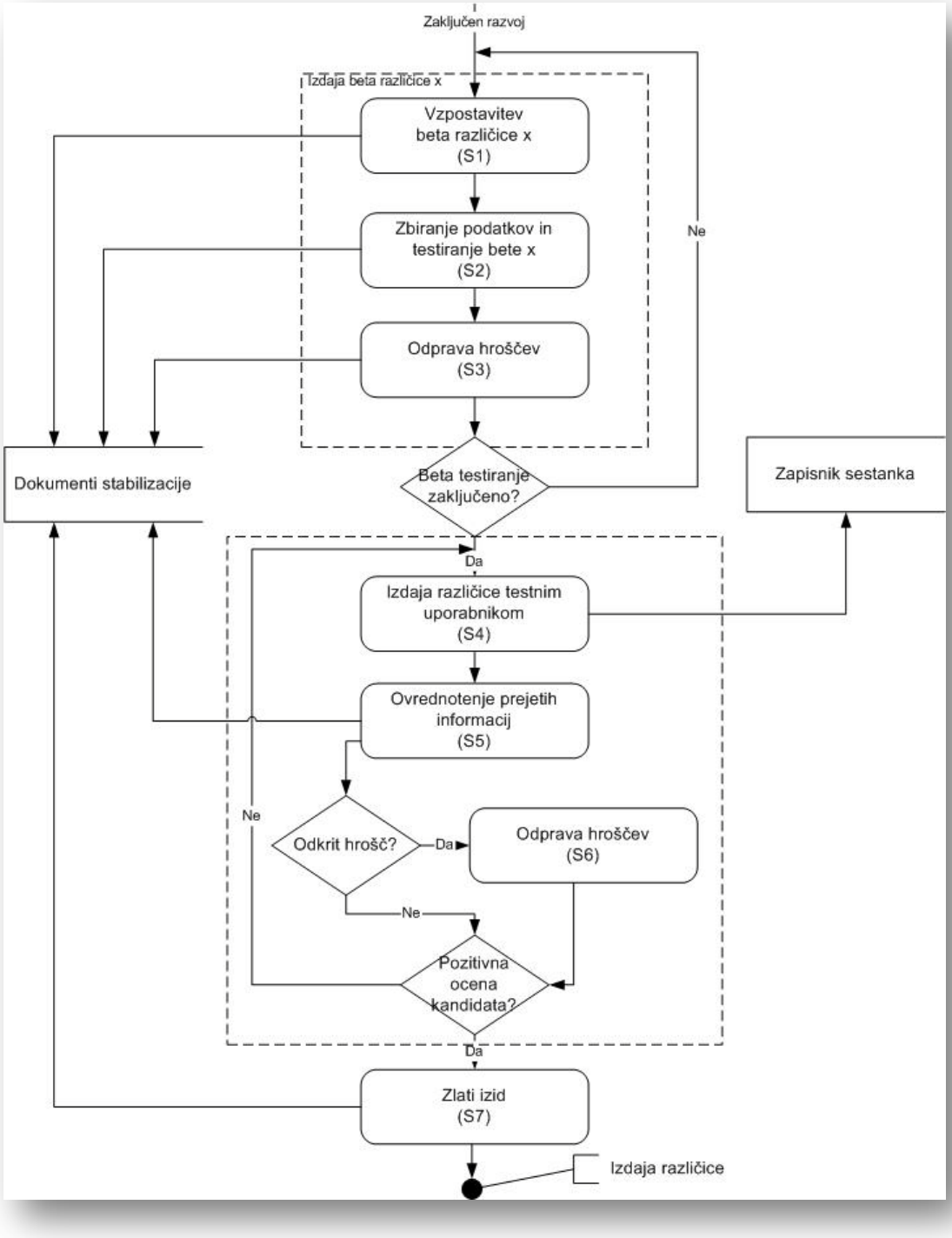
Tipično produkta ni možno zaključiti, če je nivo resnosti in prioritete 1.
Odprava hrošča je notranji korak v smeri zaključka, ki je dosežen po potrditvi testerja, da odprava hrošča ni povzročila drugih težav.
Hrošči so tipično odpravljeni s statusi:
- **odpravljen**
  Razvoj hrošča odpravi, testira popravek, kodo, določi popravek končni različici, vrne poročilo nazaj osebi, ki je hrošč sporočila.
- **podvojen**
  Ponovljen hrošč že evidentiranega hrošča, vzpostavi se povezava z originalom.
- **odložen**
  Hrošč ne bo odpravljen v trenutni različici.
- **po načrtu**
  Obnašanje hrošča je namensko in je del funkcionalne specifikacije.
- **brez ponovitve**
  Razvoj ne more preveriti ponoviti pojavitve hrošča.
- **brez popravka**
  Hrošč ne bo odpravljen, ker tim odloči, da ni vreden truda. [2]

Dnevni prevod
Čeprav je lahko težaven za izvedbo, nudi velike koristi. Predstavlja enostaven prevod celotne kode aplikacije v izvršilno datoteko. Čeprav se imenuje dnevni, pa je uporaben tudi, če ga izvajamo na vsake tri ali štiri dni. Moč dnevnega prevoda predstavlja soočenje celotnega tima z napredkom posamezne ekipe, kar pomeni dodatno stimulacijo članom. Prav tako se že takoj vidijo težave integracije, ki se jih tako zaveda celoten tim. [2]

3.4. Stabilizacija

3.4.1. Splošno
Ko tim zaključi s kodiranjem izvorne kode in doseže mejnik *zaključen razvoj*, je naloga stabilizacije, da produkt dostavi stranki. V *fazi stabilizacije* korakov, podobnih prejšnjim fazam, ni. Namesto tega v *fazi stabilizacije* tim izda več različic produkta oziroma mora doseči več internih mejnikov vse do končnega produkta. Ti mejniki so odprava hroščev, uskladitev vseh izdelkov produkta, izdaja različice in izdatno testiranje izdane različice. Pri teh vmesnih oziroma mejnih različicah, ki vse ponavljajo enake korake, se tim omeji predvsem na testiranje s ciljem ugotavljanja hroščev in njihovo odpravo. Pomemben interni mejnik, ki naznanja bližanje končnemu produktu, je različica brez hrošča (RBH ali ZBR). To je prvi interni mejnik, kjer so bili vsi aktivni hrošči obravnavani (odpravljeni, podvojen, odložen, …).
Faza stabilizacije tudi pomeni dokončanje vseh dokumentov podpore. Ti morajo prav tako biti testirani in zaključeni, preden je izdana končna različica. Šele, ko je pripravljena vsa koda

aplikacije kakor tudi ostali dokumenti, tim doseže cilj izdelave pravega produkta ob pravem času.
Stabilizacija predstavlja približno 10 do 15% celotnega projekta. [2]



Slika 7: Shema stabilizacije

3.4.2. Kdo jo izvede?

Spodnja tabela opisuje tipične zadolžitve vlog v *fazi stabilizacije*. Vodja vsake vloge skrbi za izvršitev zadanih nalog (glavna naloga je izdaja različice) in komunikacijo z ostalim člani tima.

Tabela 8: Tipične zadolžitve vlog v fazi stabilizacije

Vloga	Zadolžitve
Produktno upravljanje	Koordinira izdaje internih različic s stranko. Planira splavitev produkta.
Projektno upravljanje	Upravlja z beta in pilotskimi različicami produkta. Vzdržuje projektni urnik.
Razvoj	Omejeno na iskanje in odpravo hroščev. Zagotavlja, da je integracija produkta končana in uspešno testirana.
Uvajanje	Zagotovi dokončanje materialov za podporo uporabnikom. Določa in koordinira predavatelje in uporabnike za izobraževanje internih različic ter končnega produkta.
Testiranje	Izvede testiranje po planu. Najde, zabeleži in klasificira hrošče. Zaključi hrošče in zagotovi, da so ustrezno odpravljeni. Usmeri se tudi na testiranje uporabnosti, namestitve in konfiguracije.
Logistično upravljanje	Skrbi za namestitev, konfiguracijo, dostavo in podporo internih različic. Planira dostavo končnega produkta. Nudi podporo sistemski skupini in skupini za podporo ter skupini pomoč uporabnikom.

[6]

3.4.3. Metode dela

Kakor je bilo že omenjeno, *faza stabilizacije* ne poteka v korakih, podobnih prejšnjim fazam. Faza stabilizacije poteka preko internih mejnikov, ki jih mora tim doseči. Vsak interni mejnik predstavlja svojo različico produkta. Te interne različice si sledijo zaporedoma vse do končnega produkta. Pri vsakem od omenjenih internih mejnikov tim integrira in uskladi vse izdelke produkta. Z vsako interno različico je produkt testiran, odpravljeni so hrošči in izvedeni so popravki. Izdaja končnega produkta je skupna odločitev vseh vodij vlog, stranke ter sistemske skupine in skupine za podporo. Interni mejniki znotraj faze stabilizacije: *zmanjšanje hroščev, različica brez hroščev, kandidat za izdajo* in *končna izdaja produkta.* [2]

3.4.3.1. Iskanje hroščev

Tim lahko izda več internih različic, ki jih da v testiranje skupini uporabnikov in s tem omogoči še dodatno testiranje že v fazi razvoja. V fazi stabilizacije bi morali zabeležiti negativen trend zabeleženih hroščev, kar nakazuje vse večjo stabilnost aplikacije. [6]

3.4.3.2. Različica brez hroščev (RHB)

To je prva točka v projektu, kjer razvojni tim dohiti tim za testiranje. Tu ni več aktivnih hroščev ali pa noben ni več aktiven dlje časa (normalno 72 ur). Pričakovano je, da bo število hroščev po tej izdaji produkta naraslo, toda razvojni tim se bo trudil doseči RBH v naslednjih internih različicah. Prispetje do RHB je jasen znak, da je tim v zadnjih stopnjah projekta pred izdajo produkta. [6]

3.4.3.3. Kandidat za izdajo

Ko se tim odloči, da je produkt potencialno pripravljen za izdajo, se izdela kandidat za izdajo. Vsak kandidat za izdajo vsebuje vse elemente, ki so potrebni za izdajo produkta in nima aktivnih hroščev. Tim izvede intenzivno testiranje kandidata in tako odkrije še zadnje možne hrošče. Intenzivno testiranje ponudi odgovor, ali bo kandidat za izdajo postal končni produkt ali pa mora tim izdelati novo različico kandidata za izdajo z ustreznimi popravki. Malo verjetno je, da bi bil prvi kandidat za izdajo že tudi končni produkt. [6]

3.4.3.4. Izdaja končnega produkta

Izdaja končnega produkta je različica kandidata za izdajo, za katerega se naročnik, stranka in celoten tim strinjajo, da je to različica, ki bo predstavljala končni produkt. To je različica, ki pomeni različico za na polico, za katero se dodaten razvoj ali testiranje ne izvaja več.
Določitev, katera različica predstavlja končni produkt, je težka odločitev. Končno je odgovor na to izdaja pravega produkta ob pravem času. Ali produkt v celoti izpolnjuje zahteve uporabnika? Ostali ključni elementi so poročilo o hroščih, rezultati testiranja kandidata za izdajo in stanje podpore produktu. [7]

3.4.4. Mejnik izdaja različice ter izdelki faze stabilizacije

Dosega mejnika *izdaja različice* je končni cilj projektnega tima. Predstavlja točko, ko je tim zaključil z delom na produktu in so tako produkt, njegovi elementi in drugi spremni izdelki pripravljeni za izdajo. To je tudi točka, kjer projektni tim prepusti lastništvo, dostavo in vzdrževanje nad aplikacijo sistemski skupini in skupini za podporo. S tem je tudi zaznamovano, da je projektni tim dosegel zastavljeno vizijo projekta.
Izdelki tega mejnika nudijo pomembne informacije za dostavo in uporabo produkta v njegovem produkcijskem okolju.
Za dosego mejnika so potrebni naslednji izdelki:

- **izdaja končnega produkta**
  Izvorna koda in izvršne datoteke.
- **opombe k izdanem produktu**
  Dokument, ki vsebuje informacije o izidu in zadnjih spremembah.
- **dokumenti podpore**
  Končne različice dokumentov podpore.
- **rezultati testiranja**
  Stanje hroščev in njihova baza za kasnejše projekte.
- **projektni arhiv**
  Vse pomembne informacije o produktu, ki so bile izdelane v fazi razvoja.
- **projektna dokumentacija**
  Projektna dokumentacija iz vseh faz projekta vključno z večjimi različicami ob večjih mejnikih. [1]

3.4.4.1. Opombe k izdanemu produktu

Vsak produkt vsebuje spremembe narejene v zadnjem hipu oz. druge elemente s katerimi morajo biti stranka, uporabniki in njihova podpora seznanjeni. S pripravo enostavnega dokumenta z omenjeno vsebino, se izognemo problemom, ki lahko nastanejo ob dostavi produkta s strani tima za dostavo. [1]

3.4.4.2. Uporabniški dokumenti

Ti so lahko različnih oblik od datotek pomoči, čarovnikov, vodičev za uporabo pa do priročnikov za tečaj. Osnovni namen teh dokumentov je dvojni, priprava uporabnikov na izdajo produkta in pomoč uporabnikov po njej. [1]

3.4.4.3. Projektni dokumenti

Ključ za dobro sprejetje izdanega produkta je ustrezna dokumentacija, namenjena uporabnikom. Projektni tim lahko vnaprej objavi skorajšnji zaključek projekta in zainteresiranim uporabnikom ponudi lokacijo, kjer lahko najdejo nekatere predhodne dokumente.
Dobra ideja pred pripravo kompletne dokumentacije je predhodno priprava dokumenta, ki poudarja glavne lastnosti aplikacije. Ko se projekt nahaja na četrtini procesa dostave lahko

tim na spletni strani objavi seznam pogostih vprašanj z odgovori. Seznam naj bo redno osvežen in v njegovo pripravo je potrebno vključiti uporabnike s svojimi dejanskimi primeri. [1]

3.4.4.4. Tečaj in priročniki za tečaj

Kot del procesa stabilizacije lahko projektni tim izdela tudi plan tečaja skupaj z ustreznim priročnikom. Tečaj je lahko izdelan v obliki samostojnega učenja, formalnega tečaja z več uporabniki ali pa neformalnega tečaja z eno osebo, odvisno od velikosti dostave in zapletenosti aplikacije. [1]

3.4.4.5. Dokumenti podpore

Sistemska dokumentacija je lahko sestavljena iz več delov, odvisno od aplikacije. Dokumentacija strežnika je potrebna, če je aplikacija sestavljena iz SUBP in podatkov. Ti dokumenti so lahko dodatno deljeni v dokumentacijo o spremembah in dopolnitvah k namestitvi, sistemskih nastavitvah in sprotnih nastavitvah (primer bi bil dokument, ki opisuje obnovo podatkov po podatkovni nesreči).

Odvisno od aplikacije, predvsem če je ta izdelana večnivojsko, je potrebno dokumentirati tudi možnosti nastavitev, parametrov in metodologij, ki naj bodo predhodno testirane v internih različicah.

Prav tako je potrebno izdelati dokumentacijo odjemalcev, ki vsebuje specifikacijo namestitve aplikacije in opravljene privzete nastavitve. Dokument naj bo pripravljen že v *razvojni fazi* in zaključen že v *fazi stabilizacije*. Poleg tega je za več nivojske aplikacije smiselno izdelati okvirno skico komunikacije med nivoji. [1]

3.4.4.6. Rezultati testiranja

Zbrati in trajno arhivirati je potrebno vse rezultate testiranja aplikacije. Kasneje lahko te dokumente uporabimo za določitev novih značilnosti produkta za novo različico. [1]

3.5. Dostava produkta

Ko je produkt končan in brez hroščev, ga predamo stranki. [2]

3.6. Programska orodja, ki pomagajo pri razvoju aplikacij

Za celoten proces nastanka aplikacije potrebujemo raznovrstna programska orodja. Dandanes si je brez njih nemogoče predstavljati celotni proces razvoja. Z njimi si olajšamo delo.

Programska orodja:

- Microsoft Visio,
- Microsoft Expression Blend,
- Microsoft Project,
- Microsoft Visual Studio.

3.6.1. Microsoft Visio

Microsoft Visio je programsko orodje, ki je namenjeno ustvarjanju diagramov (UML), gradnji podatkovnih modelov in je v pomoč pri risanju grafov. Z dinamični vizualnimi učinki, ki temeljijo na podatkih, si poenostavimo zapletenost. [8]

Podobni programi:

- Kivio ,
- IBM Rational Rose,
- Dia,

- ArgoUML ,
- StarUML .[9]

3.6.2. Microsoft Expression Blend

Expression Blend je profesionalno oblikovalsko orodje za izdelavo privlačnih namiznih in s spletom povezanih uporabniških izkušenj za okolje Windows. Z njim lahko izdelujemo prototipe aplikacij kot tudi končne aplikacije. [10]

Podobni programi:

- Visual Studio 2010,
- Sothink Quicker for Silverlight.

3.6.3. Microsoft Project

Informacijska podpora s programom Microsoft Project vodji projektov omogoča nazoren pregled nad projekti. Z diagrami postanejo zasedenost delavcev in njihove naloge v trenutku jasne. Bistveno se skrajša čas načrtovanja, izvajanja in kontrole projektov. Za učinkovit nadzor nad potekom dogodkov je dovolj le pogled, tudi v primeru dodatnih sprememb. Z večanjem obsega in števila projektov sicer narašča možnost za neporazdeljene obremenitve virov, vendar Project omogoča njihov optimalni izkoristek ali skrajšanje časa izvedbe projekta. Na predvidene aktivnosti nas redno opozarjajo opomniki. [11]

Podobni programi:

- OpenProj,
- GanttProject,
- Open Workbench. [12]

3.6.4. Microsoft Visual Studio

Microsoftov razvojni sistem Visual Studio je zbirka razvojnih orodij, ki razvijalcem programske opreme, pa naj gre za začetnike ali izkušene strokovnjake, pomaga premagovati kompleksne izzive in ustvarjati inovativne rešitve. Vloga zbirke Visual Studio je izboljšati razvojni proces ter omogočiti preprostejše uresničevanje inovacij. [13]

Podobni programi:

- Qt Creator,
- Code::Blocks,
- SharpDevelop,
- MonoDevelop. [14]

4. Expression Blend 3 – enostavna rešitev za izdelavo prototipov

Microsoft Expression Blend 3 je orodje, namenjeno oblikovalcem za gradnjo privlačnih (namiznih WPF ali spletnih Silverlight) aplikacij. Orodje je dokaj novo, zato je na začetku potrebna dodatna krivulja učenja, da lahko začnemo graditi smiselne in bogate aplikacije. Veliko razvijalcev in oblikovalcev se tako zateče k svojemu priljubljenemu ponudniku iskanja za raziskovanje primerov in navodil, kako se lotiti razvijanja ali oblikovanja v tem orodju.
Microsoft Expression Blend 3 (kot tudi veliko drugih Microsoft produktov) vsebuje veliko primerov uporabe ob zagonu orodja. Na začetnem oknu (Startup Window) ponuja precej privlačnih primerov uporabe, vodičev, ki se jih da izkoristiti. [15]
WPF in Silverlight zagotavljata platformo, ki prinese bogato, animirano, s predlogami aplikacijo. Zagotavljata zapleteno vektorsko grafiko in briše razliko med uporabniškim vmesnikom in vsebino. Ustvarimo lahko aplikacije s standardnim izgledom in si s tem poenostavimo razvoj. Na ta način dobimo več prožnosti in enostavnosti. Animacija in povezava s podatki sta možni skoraj povsod, tako da je aplikacija interaktivna. Lahko se uvozijo slike, video, 3D. Expression Blend je tudi v celoti napisan v WPF-ju. WPF je SDK, razvojna programska oprema, ki ni prijazna uporabniku. Tukaj pa pride v poštev Expression Blend. Blend je vizualno orodje za WPF in Silverlight in je oblikovalsko prijazno orodje, v katerem lahko uporabljamo celoten WPF. Omogoča risanje, animiranje, dostop do komponent in kontrol, povezovanje z bazo podatkov. Ima še veliko drugih funkcij, ki pripomorejo k boljši uporabniški izkušnji. Rišemo lahko tudi s pomočjo tablic za risanje. Uvažamo lahko slike, ki so v Photoshop formatu.
Kompleksne aplikacije imajo ponavadi veliko kode in funkcionalnosti. Expression Blend nudi popolno podporo popravljanja kode XAML, C# in vb. Na voljo je tudi »intellisense«, s katerim lažje programiramo. Projekt, ki ga ustvarimo v Expression Blendu, je možno odpreti tudi z Microsoft Visual Studiem. Možna je pretvorba prototip aplikacije v neko osnovo za nadaljevanje projekta v okolju Microsoft Visual Studio.

Slika 8: Expression Blend.

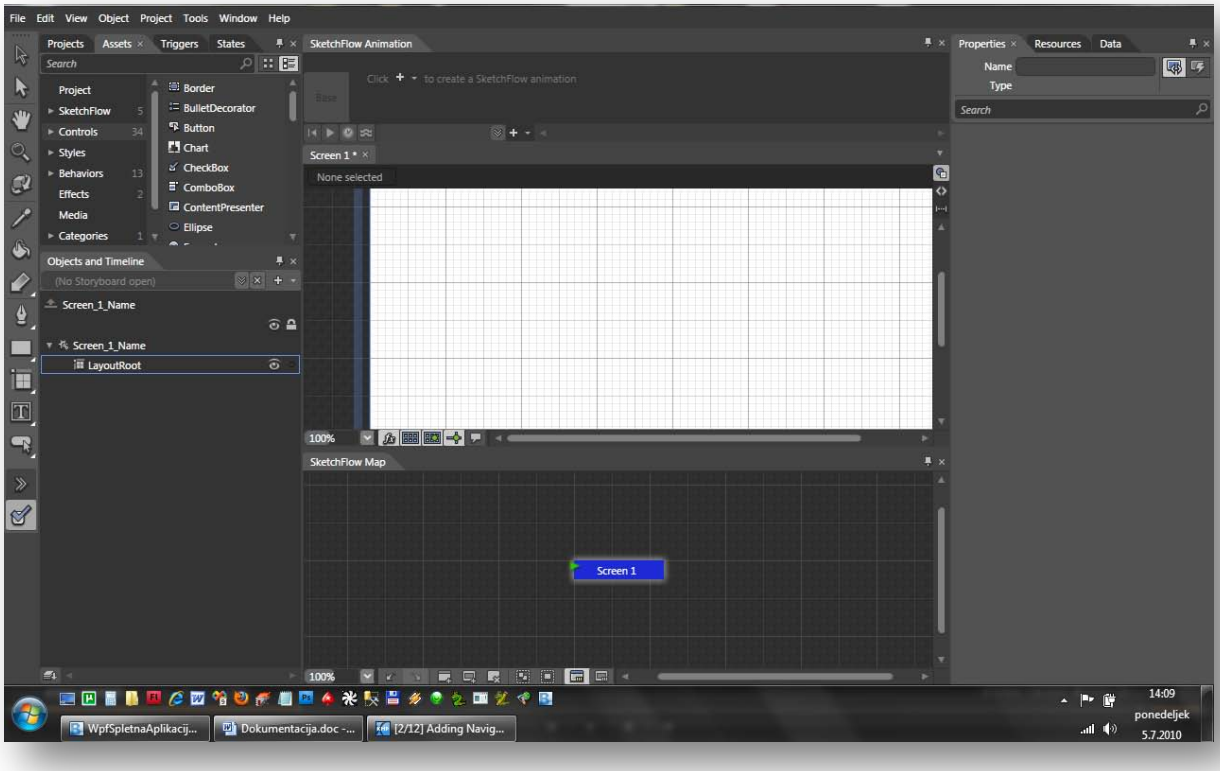
4.1. Expression Blend 4 in njegove novosti

Novosti v Expression Blend 4:

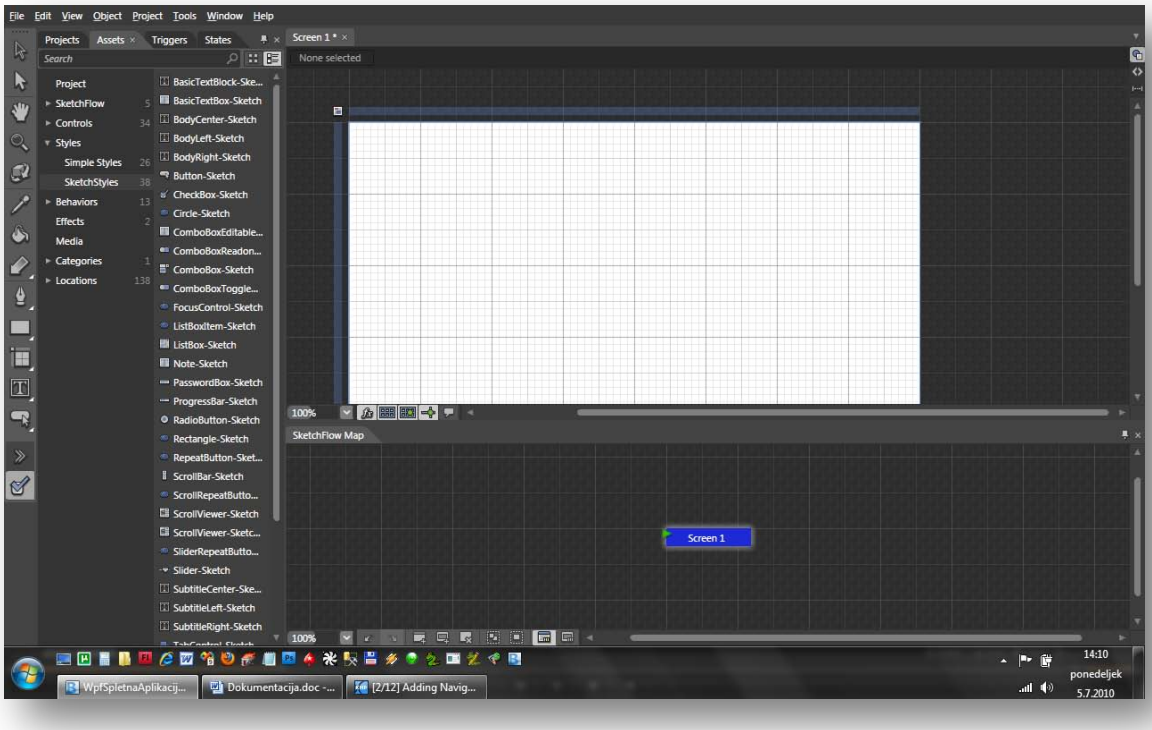
- Kompatibilnost s Silverlight 3 in WPF 3.5 z servisnim paketom (SP1),
- možnost objavljanja v Microsoft SharePoint,
- pretvorba povratnih informacij (feedback) v TFS delovno nalogo,
- menjavanje SketchStyle,
- možnost pavziranja in nadaljevanje SketchFlow animacij,
- izboljšano uvažanje Photoshop slik,
- nove kontrole,
- izboljšano oblikovanje kontrol,
- manj XAML kode. [16]

4.2. Prvi stik z novim okoljem in ureditev okolja na delo

Kreiramo nov projekt tako, da izberemo File→ New Project → »Wpf SketchFlow Application« oz. »Silverlight SketchFlow Application«. Vnesemo želeno ime in pritisnemo »OK«. Odpre se nam prazno okno. Prva stvar, ki jo naredimo, je ta, da spremenimo izgled delovne površine. To naredimo tako, da gremo na Window → Workspaces → Design in nato Windows → »Reset Current Workspace«, tako da dobimo osnovni delovni prostor. Nato si odstranimo tiste zavihke, ki jih ne potrebujemo. V našem primeru odstranimo »Object and timeline«, »SketchFlow Animation«, »Properties«, »Resources«, »Data«. Namen tega je, da si pustimo odprte samo tiste zavihke, ki jih trenutno potrebujemo. S tem si ustvarimo večjo delovno površino. S tem, ko zapremo določene zavihke, ni nič narobe, saj jih lahko vedno naknadno prikažemo. To storimo tako, da na meniju kliknemo »Windows« in tam imamo na voljo za obkljukati zavihke, ki jih potrebujemo. Ko si vse priredimo po naših željah, lahko delovno površino shranimo, tako da gremo na Window → »Save as New Workspace«, ga poimenujemo in shranimo. Tako ga lahko uporabimo tudi pri drugih projektih in ne zgubljamo časa s ponovnim nastavljanjem. Obstaja pa tudi bližnjica, ki navidezno zapre vse zavihke in tako nam ostane samo delovna površina. Ta bližnjica je tipka »F4«, ki nam prikaže (maksimira) oziroma skrije (minimira) zavihke. [17]



Slika 9: Delovno okolje pred popravki.

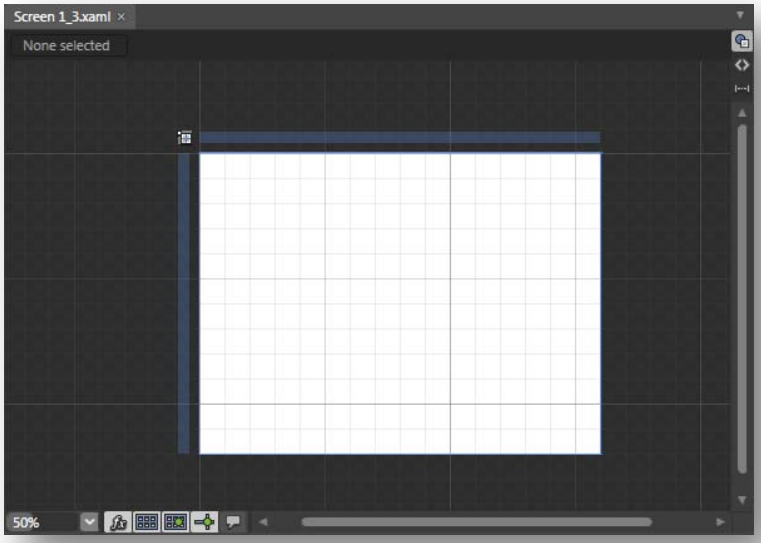


Slika 10: Delovno okolje po popravkih.

4.2.1. Predstavitev zavihkov

Obdelovalno okno

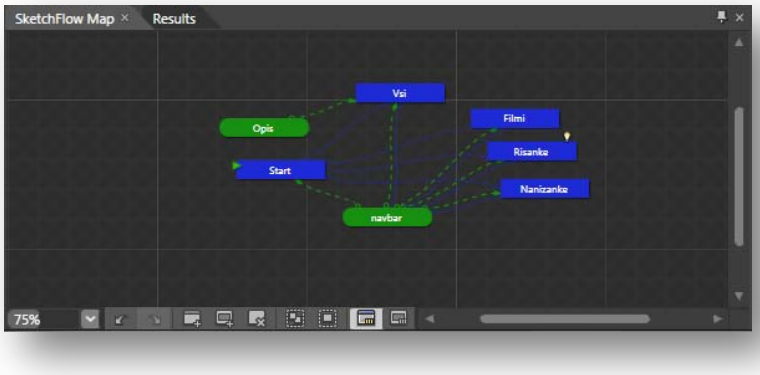
Kot pove že samo ime, tukaj prikazujemo želene podatke.



Slika 11: Zavihek Obdelovalnega okna.

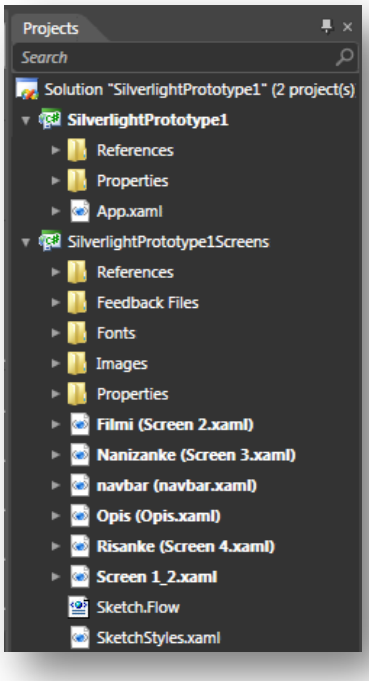
SketchFlow map

V SketchFlow map vidimo naš projekt v obliki nekakšnega miselnega vzorca. Imamo okna in komponentna okna, ki so povezana med sabo. Te povezave nam kažejo, kaj je med seboj povezano. S klikom na katerokoli okno oziroma komponentno okno se nam le to odpre.



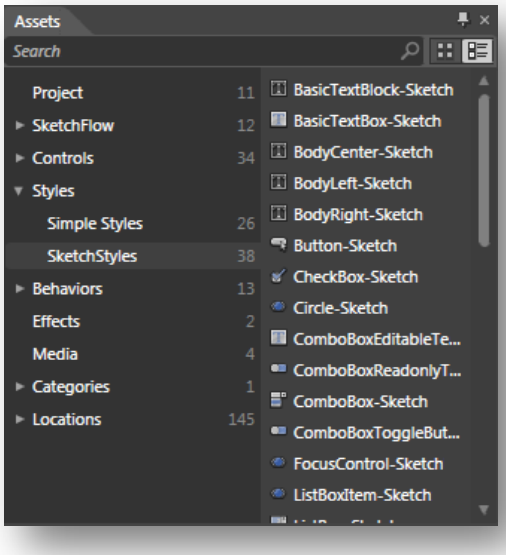
Slika 12: Zavihek SketchFlow Map.

Projects
Tukaj lahko neposredno dostopamo do vseh datotek, ki jih imamo v projektu.



Slika 13: Zavihek Projects.

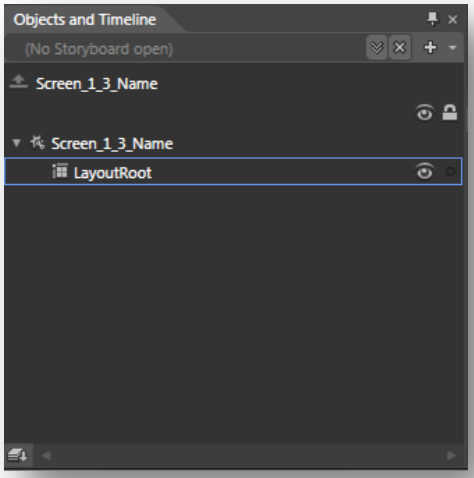
Assets
V Assets imamo na voljo veliko različnih kontrol za delo v Expression Blendu.



Slika 14: Zavihek Assets.

Object and Timeline

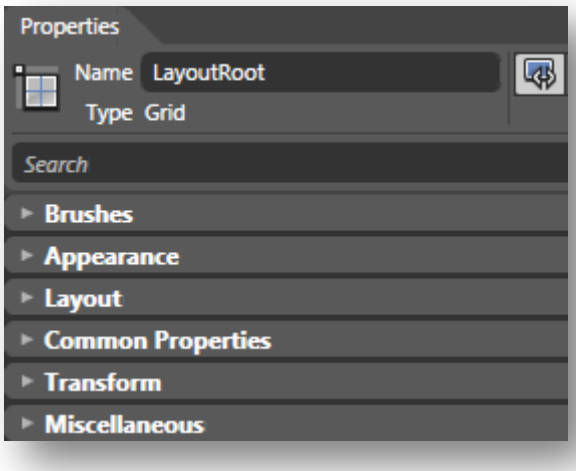
Tukaj imamo seznam kontrol, ki jih imamo trenutno na obdelovalnem oknu, in jim lahko nastavljamo različne lastnosti.



Slika 15: Zavihek Objects and Timeline.

Properties

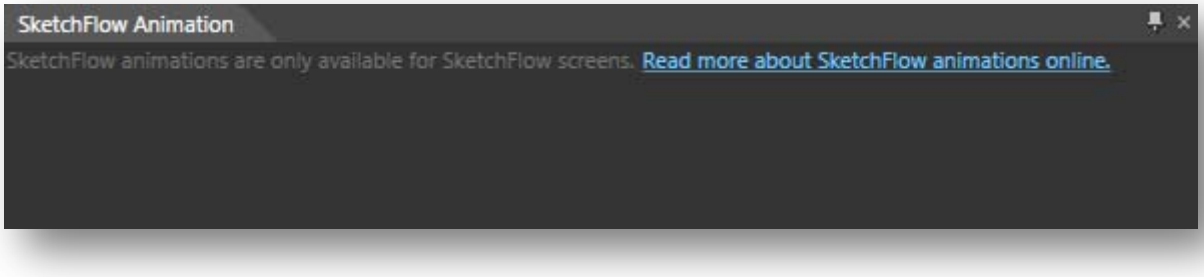
V Properies nastavljamo kontrolam razne lastnosti, akcije, stanje.



Slika 16: Zavihek Properties.

SketchFlow Animation

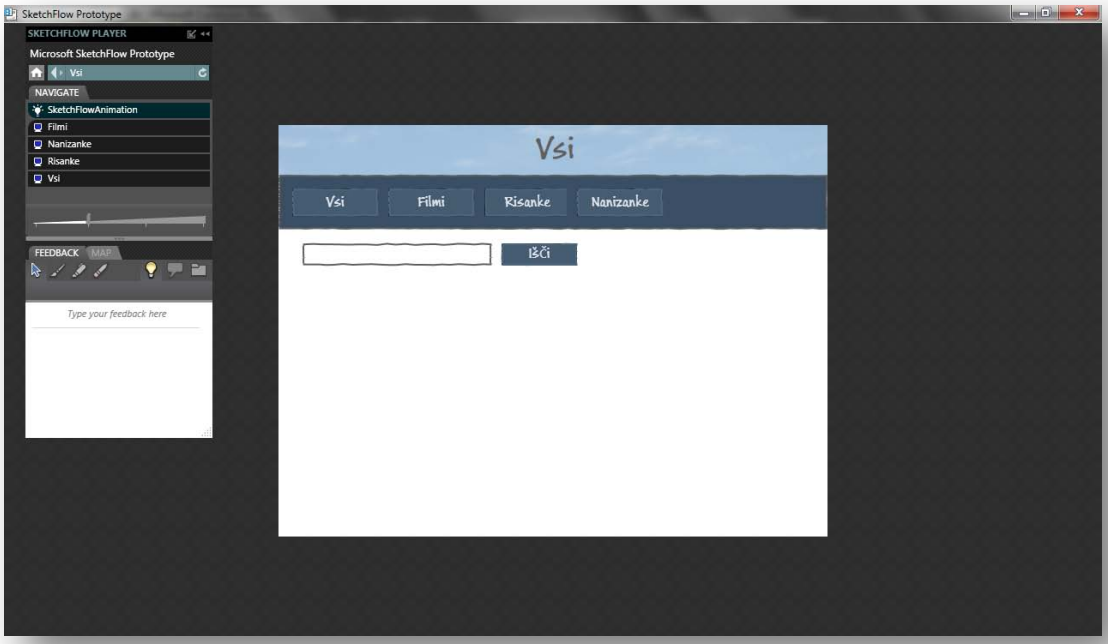
SketchFlow Animation je namenjen za izdelavo enostavnih animacij.



Slika 17: Zavihek SketchFlow Animation.

V Expression Blendu imamo še veliko pomembnih stvari, ki jih bomo spoznali tekom tega poglavja. Spoznali bomo še veliko drugih stvari kot so naprimer Feedback, Data, ki jih bomo prikazali tekom primera izdelave preproste spletne aplikacije.

4.3. Predstavitev SketchFlow player-ja

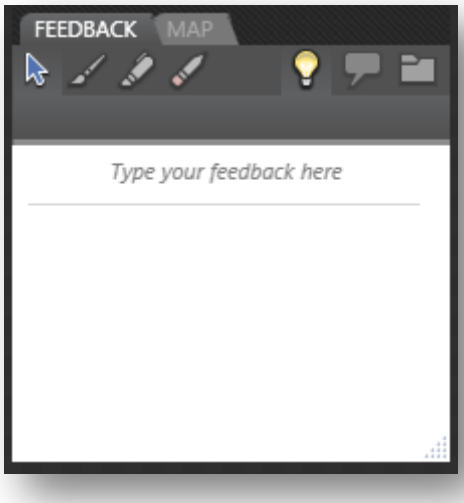


Slika 18: Predvajalnik SketchFlow player.

Ko zaženemo projekt (tipka F5), se nam odpre SketchFlow player. Znotraj player-ja lahko uporabnik navigira skozi različna okna. Z uporabo navigacijskega okna lahko uporabnik uporabi SketchFlow map za premikanje po različnih oknih.

Slika 19: Navigacijsko okno v predvajalniku SketchFlow player.

SketchFlow-player omogoča, da uporabnik daje razne povratne informacije s pomočjo ikone svinčnika ali označevalnika. To lahko shrani in ustvari se datoteka s končnico ».feedback«. Datoteko lahko pošlje želeni osebi in na ta način je komunikacija hitra. Kot primer lahko vzamemo izvajalca in stranko.

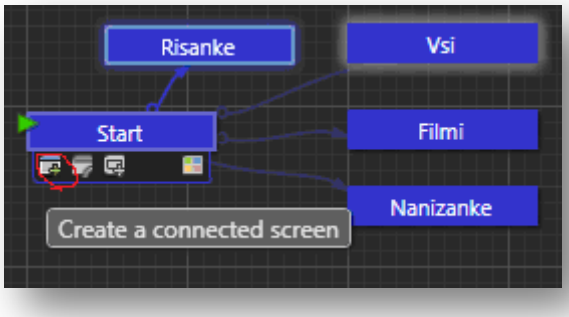


Slika 20: Feedback v SketchFlow playerju.

4.4. Izdelava prototipa preproste spletne aplikacije

4.4.1. Kratek opis aplikacije

Za izdelavo preprostega prototipa spletne aplikacije si bomo izbrali aplikacijo, kjer bomo imeli bazo filmov, nanizank in risank. Na voljo bomo imeli iskanje po vseh ali pa po posameznem atributu. Po izbiri določenega filma se nam bo prikazal poster tega filma, opis in možnost ogleda trailerja. Za izdelavo tega prototipa si bom pomagal z orodjem Expression Blend 3 in pomočjo Silverlight platforme.



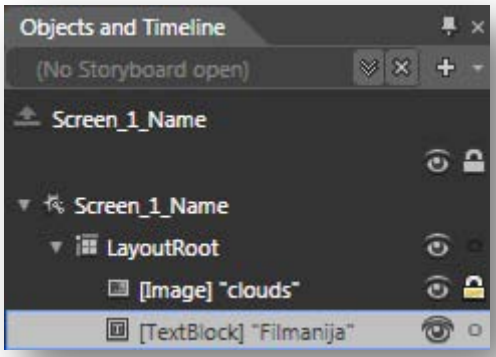
Slika 21: SketchFlow Map in nodi.

4.4.2. SketchFlow map in nodi

Najprej se lotimo dodajanje nodov (strani, oken). Posamezen node predstavlja svojo stran. Zato dodamo node (strani), ki jih bomo potrebovali in jih ustrezno poimenujemo.

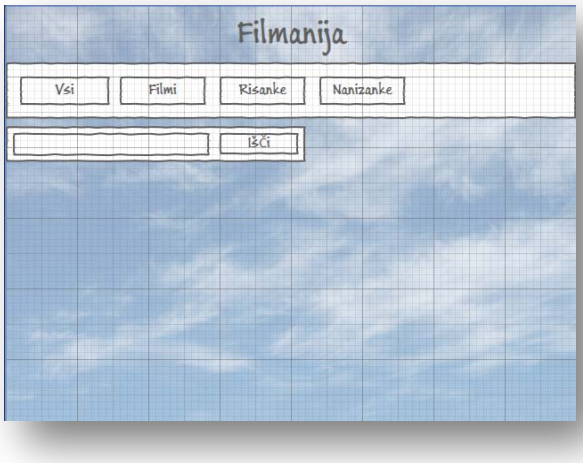
Kliknemo na vsak node in napišemo naslov. Na node »Start« želimo najprej dodati ozadje. Za to je potrebno klikniti na Project → »Add Existing item« in poiskati sliko, ki jo želimo

uporabiti za ozadje. Če slika po uvozu še ni lepo poravnana, jo je potrebno poravnati tako, da se prilega ozadju. Sliko bomo uporabili za ozadje, zato jo želimo nastaviti kot statično, da ne bi prišlo do neželenih premikov. Na voljo imamo možnost zaklepanja slike. To naredimo tako, da izberemo sliko in z desnim miškinim gumbom kliknemo in izberemo Order → »Send to Back«. Sliko lahko zaklenemo tudi na drugačen način. V zavihku »Objects and Timeline« vidimo vse, kar imamo trenutno na obdelovalnem oknu. Poiščemo sliko in kliknemo ikono kroga. To povzroči, da se slika zaklene (ikona ključavnice). Postopek ponovimo za vse strani, kjer želimo imeti takšno ozadje.



Slika 22: Kontrole, ki so trenutno na izbranem oknu.

Naslednja stvar, ki se je lotimo je dodajanje navigacijskih gumbov. Še preden se lotimo gumbov, pa prenesemo ozadje za gumbe. To najlažje storimo tako, da gremo na zavihek Assets → Syles → SketchStyle in izberemo »Rectangle-Sketch«. Na izbiro imamo možnost, da kliknemo in povlečemo kontrolo na okno. Lahko pa kontrolo izberemo in jo na oknu narišemo v poljubni velikosti (pravokotnik). Nato v ta pravokotnik povlečemo »Button-Sketch« (gumb). Za hitro kopiranje gumba samo držimo tipko »Alt«, kliknemo na gumb in ga premaknemo na želeno mesto. Tako ostane prvotni gumb na svojem mestu, kopirani gumb pa tam, kamor ga premaknemo. To ponovimo za vse gumbe. Na podoben način ustvarimo tudi polje za iskanje po bazi filmov. Dodamo »Rectangle-Sketch«, znotraj njega pa gumb Išči in »TextBlock-Sketch«.



Slika 23: Okno filmanija.

4.4.3. Komponentna okna

Če želimo imeti iskalnik in navigacijske gumbe na vseh straneh, jih lahko pretvorimo v kontrolo in to komponentno okno uporabimo na več straneh. Če so potrebni popravki, jih

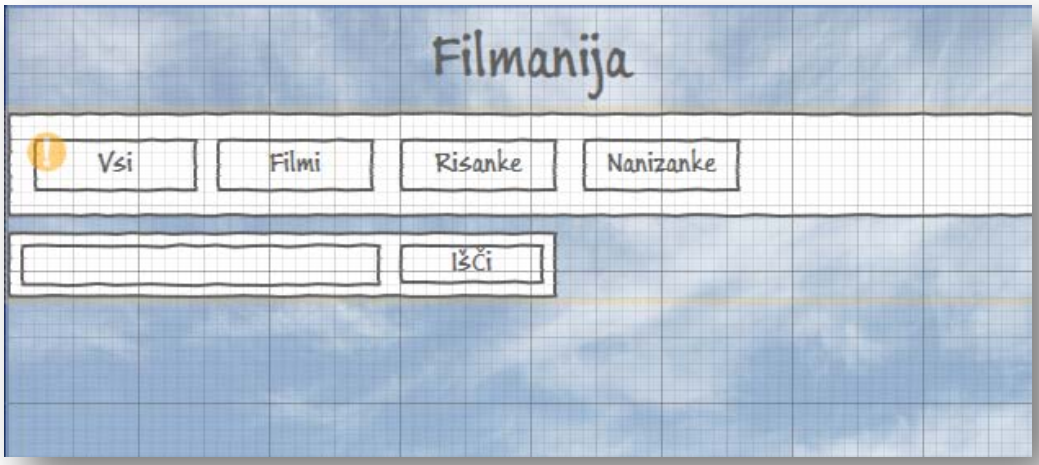
lahko naredimo samo na mestu, kjer se dejansko nahaja kontrola. Tako se izognemo napakam in skrajšamo čas popravljanja.

4.4.4. Kako do komponentnega okna

Najprej označimo vse elemente, ki jih želimo imeti v komponentnem oknu. V našem primeru označimo navigacijsko polje in iskalnik (glej zgornjo sliko). Z desnim klikom med možnostmi izberemo »Make Into Component Screen…«, poimenujemo in potrdimo datoteko. Ustvari se nam nov dokument (navbar.xaml). V »sketchflow map« vidimo novo ikono »navbar«, ki se razlikuje od ostalih. Običajna okna imajo obliko pravokotnika, komponentna okna pa so takšne oblike, kot je prikazano na spodnji sliki.

Slika 24: Ikona komponetnega okna navbar.

Če želimo uporabiti to komponetno okno »navbar«, ga izberemo in povlečemo na vsa okna, kjer ga potrebujemo. Tako naredimo črtkane povezave do oken. Komponentno okno prepoznamo na navadnih oknih po ikoni klicaja kot je razvidno iz spodnje slike.



Slika 25: Komponetno okno navbar na oknu filmanija.

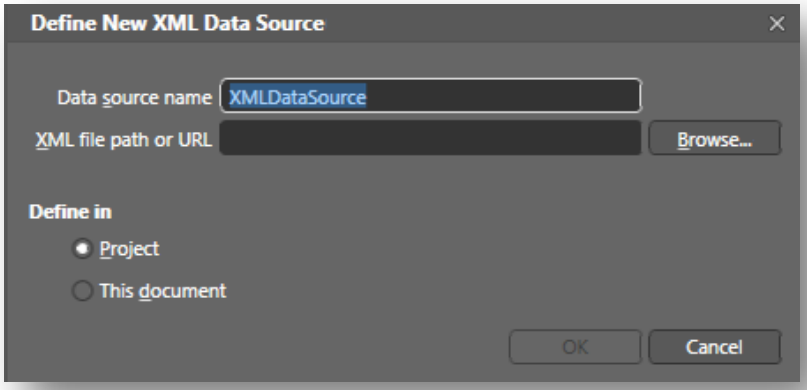
4.4.5. Gumbi in akcije

Če želimo, da imajo gumbi dejansko neko uporabnost nas mora posamezen gumb presmeriti na želeno okno. To naredimo tako, da odpremo komponentno okno »navbar«. Pred nami se odpre komponentno okno z navigacijskimi gumbi. Označimo gumb »Vsi« in kliknemo desni gumb na miški. Z miško gremo na »Navigate to« in izberemo »Vsi«. S tem, ko smo to izbrali smo dodali akcijo na ta gumb. Kadar bomo kliknili na ta gumb, se nam bo odprla stran »Vsi«. Isto naredimo za gumb (Filmi, Risanke, Nanizanke).

4.4.6. Iskalnik in xml podatki

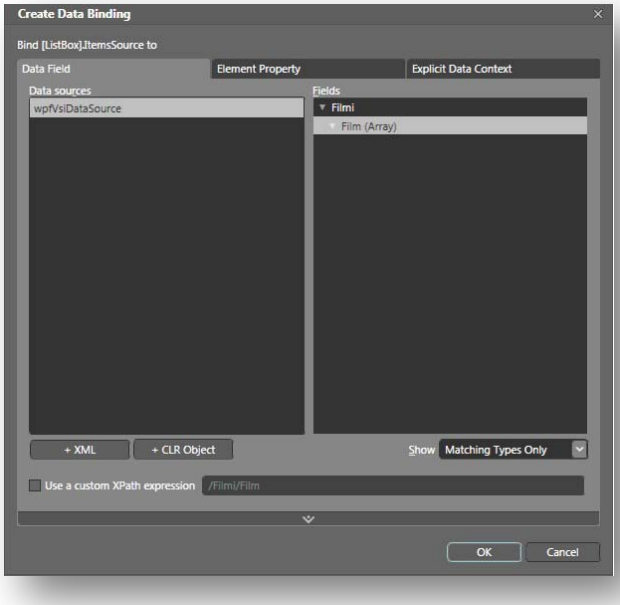
Za rezultat iskanja po filmih bomo potrebovali »ListBox-Sketch«. Ta se pa nahaja pod Assets → SketchStyles → »ListBox-Sketch«. Izberemo in povlečemo na želeno okno. Sedaj imamo

»ListBox«, ki ga želimo napolniti s želenimi podatki. To storimo tako, da označimo »ListBox« in kliknemo desni klik pri tem pa izberemo »Bind ItemSource to Data«. Kliknemo na »+xml«. Odpre se okno »Define New XML Data Source«. S klikom na »Browse…« poiščemo že vnaprej pripravljen XML napolnjen s podatki in ga izberemo.



Slika 26: Prikazno okno za izbiro xml podatkov.

Sedaj vidimo okno kot je prikazan na spodnji sliki. Kliknemo na zavihek »DataField« in pod »Data Sources« izberemo »wpfVsiDataSource«. V Polju »Field«s pa pošičemo po korenu navzdol »Film(Array)« in ga izberemo. Ko to naredimo potrdimo in naš »ListBox« bi moral prikazati podatke katere smo imeli v XML-ju.



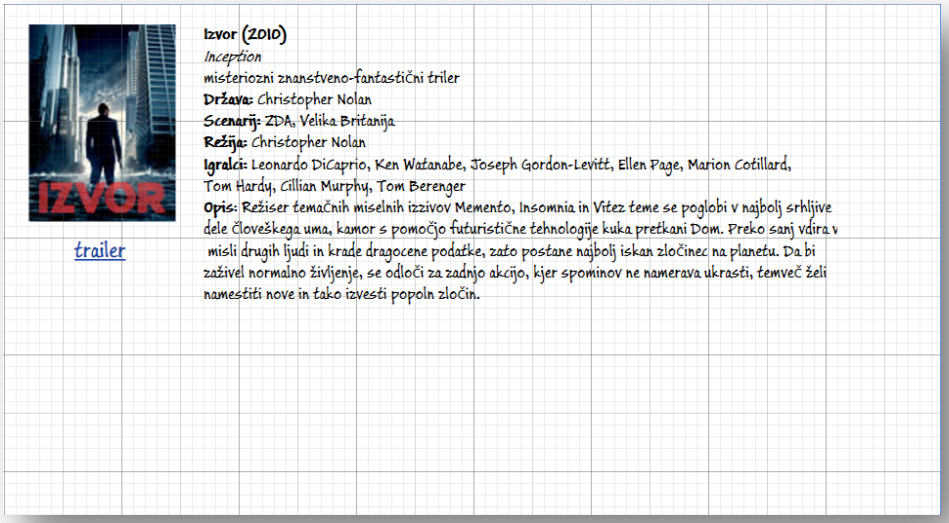
Slika 27: Prikazno okno za izbiro točno določenih podatkov.

Prikaže se nam »ListBox-Sketch«, napolnjen s podatki.

4.4.7. Okno Opis in primer opisa filma

V »SketchFlow map« dodamo novo komponentno okno. To naredimo tako, da kliknemo desni gumb na miški, izberemo »Create Component Screen« in ga poimenujemo kot »Opis«. Odpre se nam prazno okno, na katerem bomo prikazali opis filma. Gremo na »Object and Timeline« in preimenujemo okno, da dobi prijaznejše ime. Spremenimo ime

»Screen_1_5_Name« v Opis. Izberemo okno in v Properties → »Layout« nastavimo višino in širino okna na »640x350«. Sledi uvoz prej pripravljene slike (v našem primeru bomo naredili primer za film Izvor). Zatem v Assets → »SketchStyles« poiščemo »TextBlock-Sketch« in ga prenesemo na komponento. »Textblock« razširimo toliko, da je dovolj prostora za besedilo, ki ga bomo vnesli. Vnesemo želene podatke o filmu. Pod sliko vstavimo še »TextBlock-Sketch«, ki bo namenjen za prikaz trailerja. Ime spremenimo v trailer. V »Properties« mu spremenimo barvo v modro in mu damo podčrtan slog, da bo videti kot gumb.

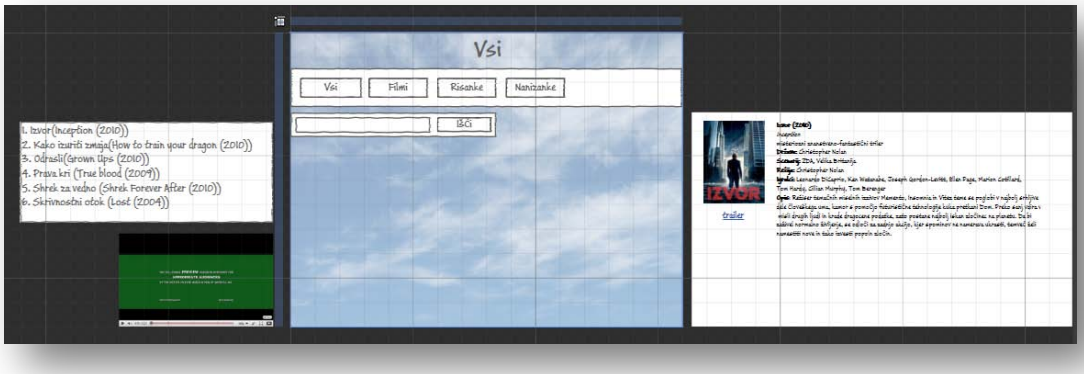


Slika 28: Okno opis, ki prikazuje opis filma.

4.4.8. Animacija iskanja in izbira filma

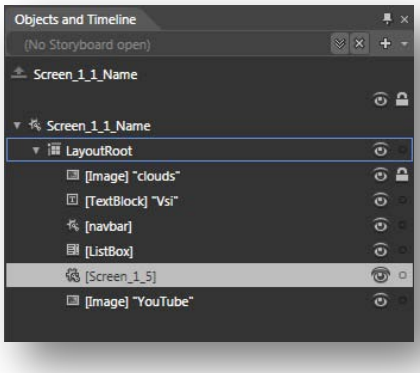
Najprej označimo »ListBox-Sketch« (rezultat iskanja) in v »Properties« pod »Apperance« nastavimo »Opacity« na 0%.

Nato povežemo komponetno okno »Opis« s stranjo »Vsi«, tako da se nam prikaže komponenta »Opis« na strani »Vsi«. Poravnamo jo tako, da nam ustreza. Tako kot pri »ListBox-Sketch« nastavimo »Opacity« na 0%. Pripnemo še sliko Youtube predvajalnika in jo premaknemo na želeno pozicijo. Nastavimo »Opacity« na 0%. Pripnemo še sliko miškinega kurzorja, vendar tukaj pustimo »Opacity« na 100%. Umaknemo sliko iz vidnega polja okna »Opis«. To naredimo zato, ker bomo glede na animacijo prikazovali in skrivali določene stvari in bomo za bolj realističen izgled premikali sliko kurzorja.



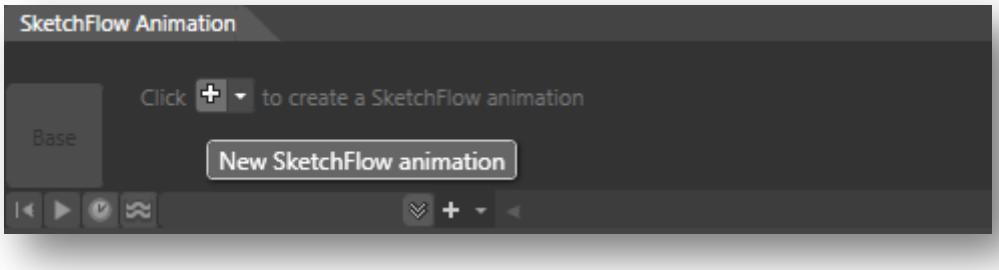
Slika 29: Kontrole, ki bojo uporabljene za animacijo.

Za kasnejše lažje izbiranje »ListBox-a« in ostalih predmetov si bomo pomagali tako, da jih bomo izbirali v »Objects and Timeline«.



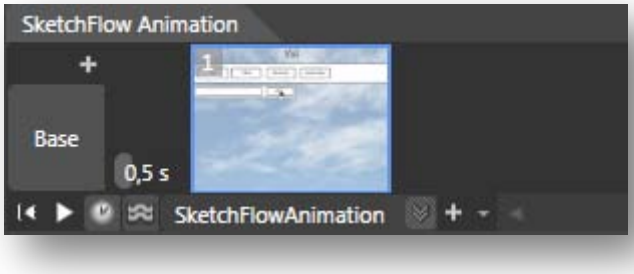
Slika 30: Kontrole, ki so trenutno na izbranem oknu.

Če še nimamo odprtega zavihka za »SketchFlow Animation«, gremo pod »Window« in izberemo »SketchFlow Animation«. Tako se nam prikaže zavihek, ki ga potrebujemo za animacijo. Kliknemo na plus.



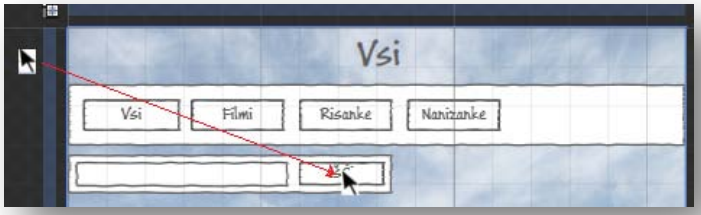
Slika 31: Prva stvar, ki jo naredimo pri animaciji.

S tem, ko kliknemo na plus, se nam odpre okence za animacijo, ki je po defaultu nastavljena na trajanje 0,5s. To lahko spremenimo po želji. Rdeča obroba in napis »SkectFlow animation Frame_1 recording is on« nas opozarjata, da imamo vključeno snemanje.



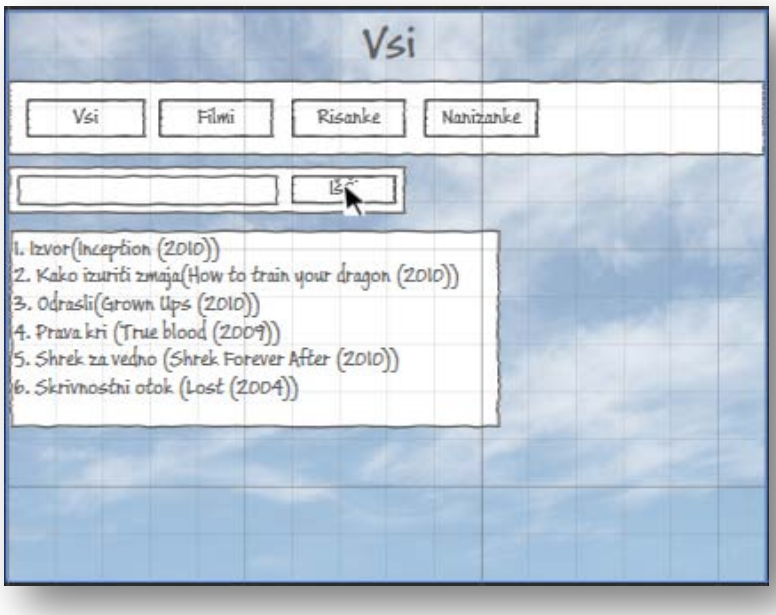
Slika 32: Prikaz slike po kliku na ikono plusa.

V naslednji potezi želimo premakniti sliko miškinega kurzorja na gumb Išči. S tem bomo ponazorili premik na gumb in klik. To je prvi del animacije.



Slika 33: Prikaz premika prvega dela animacije.

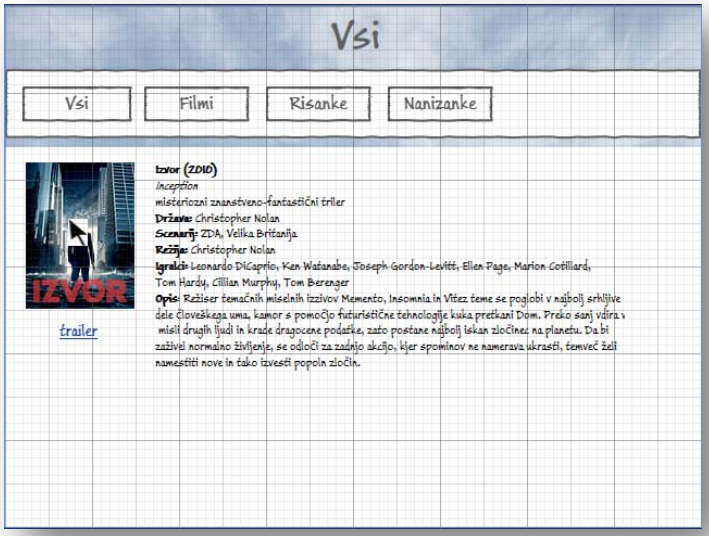
Drugi del animacije naredimo tako, da se postavimo v zavihek »SketchFlow animation«, kliknemo desni gumb na miški in izberemo »Insert New«. Sedaj lahko naredimo drugi del animacije. Gremo v zavihek »Object and Timeline« in izberemo »ListBox-Skecth« in mu nastavimo »Opacity« na 100%, da prikažemo »ListBox«.



Slika 34: Prikaz rezultata iskanja.

Tretji del animacije naredimo tako, da spet izberemo »Insert New«. Premaknemo sliko kurzorja na film Izvor in s tem smo zaključili tretji del animacije.

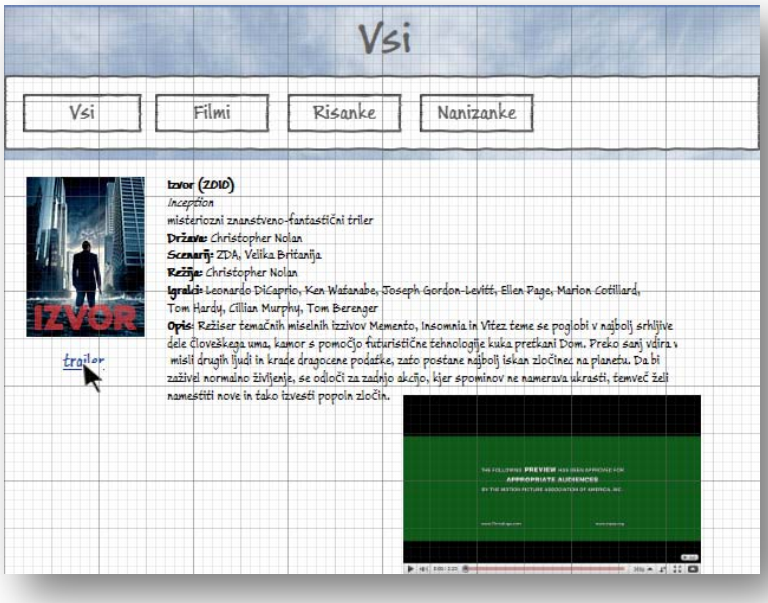
Pri četrtem delu animacije nastavimo »ListBox-Sketch« (rezultat iskanja) »Opacity« na 0% in komponentnemu oknu Opis nastavimo »Opacity« na 100%. Prikaže se nam opis filma Izvor.



Slika 35: Prikaz opisa filma.

Za peti del animacije ponovno izberemo »Insert New« in premaknemo sliko kurzorja na napis trailer.

Za šesti del animacije spet izberemo »Insert New« in sliki YouTube nastavimo »Opacity« na 100%. Prikaže se nam slika trailerja.



Slika 36: Prikaz trailerja.

4.4.9. Predloga

V Expression Blend-u imamo na voljo predloge, lahko pa vsako stvar oblikujemo sami. Lahko imamo naprimer več gumbov, ki imajo enak stil oblikovanja. Popravki so potrebni samo enkrat, da vsem gumbom spremenimo slog. Na voljo pa imamo tudi možnost, da vsak gumb spreminjamo posebej. Če želimo imeti isti slog za vse gumbe, moramo spreminjati nastavitve v točno določeni datoteki (SketchStyle.xaml). Poiščemo na primer »Button-

Sketch« in nastavimo želene nastavitve v »Properties«. V primeru, da želimo nek gumb, ki bo drugačen od ostalih, pa se lotimo spreminjanja tam, kjer smo ga prenesli na obdelovalno okno. Gremo v »Properties« in ga spremenimo po potrebi.

4.4.9.1. Spreminjanje predloge

Za spreminjanje predloge, moramo poiskati zavihek »Resources« in v »SketchStyle.xaml« imamo vse kontrole, ki jih lahko spreminjamo.

Button-Sketch

Poiščemo »Button-Sketch« in ga izberemo. V zavihku »Properties« mu najprej nastavimo širino in višino na 100 x 32. Nato nastavimo barvo gumba »Background« (#FF465C71), »BorderBrush« (#FF4E667D), »Foreground« (#FFDDE4EC).

Rectangle-Sketch

Poiščemo »Rectangle-Sketch« in ga izberemo. V zavihku »Properties« mu spremenimo »Background« (#FF524E75).

Navbar.xaml

Naknadno smo se odločili, da odstranimo »Recktangle-Sketch« pri iskalniku zaradi izgleda, zato ga odstranimo.

Ozadje

Ozadje, ki je trenutno ozadje cele strani, bi vidno samo pri naslovu, ostalo ozadje bo pa belo.

ListBox-Sketch

Za »ListBox« želimo nastaviti prosojno ozadje, na katerem se vidi besedilo. »BorderBrush« nastaviti na belo barvo.

4.4.10. Izvoz projekta in povratne informacije

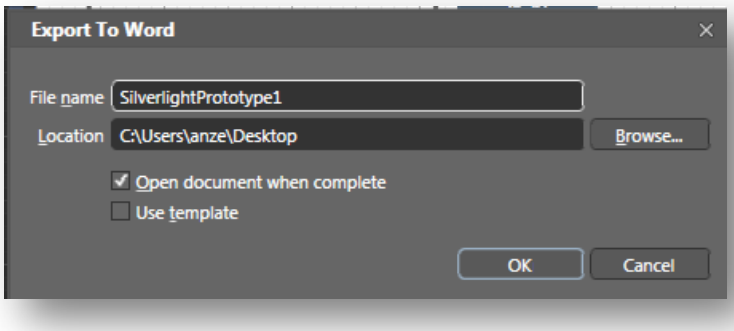
Ob zaključku projekta oziroma ob potrebi po tem, da stranko seznanimo s trenutnim stanjem projekta, lahko projekt enostavno izvozimo in pošljemo stranki.

4.4.10.1. Izvoz projekta

Za izvoz je potrebno izbrati Project → »Package SketchFlow Project«. Pokaže se nam okno, kjer imamo na voljo izbiro ime in pot, kamor naj se datoteke shranijo. Ko to potrdimo, se nam zgenerira mapa z našim projektom, ki jo stisnemo in pošljemo stranki.

4.4.10.2. Izvoz datoteke word

Na voljo imamo tudi izvoz projekta v Word datoteko. Izberemo File → »Export to Microsoft Word«. Prikaže se nam okno za ime datoteke in pot, kamor bomo datoteko shranili. Na voljo imamo, da se nam datoteka odpre takoj, ko jo shranimo in na voljo imamo tudi, da uporabimo vnaprej pripravljene predloge. Datoteka vsebuje kazalo vsebine, kazalo slik ter vse slike oken in komponent, ki jih imamo v projektu. To datoteko lahko tudi pošljemo stranki v pregled oziroma jo lahko uporabimo tudi kot nekakšen osnutek za dokumentacijo.

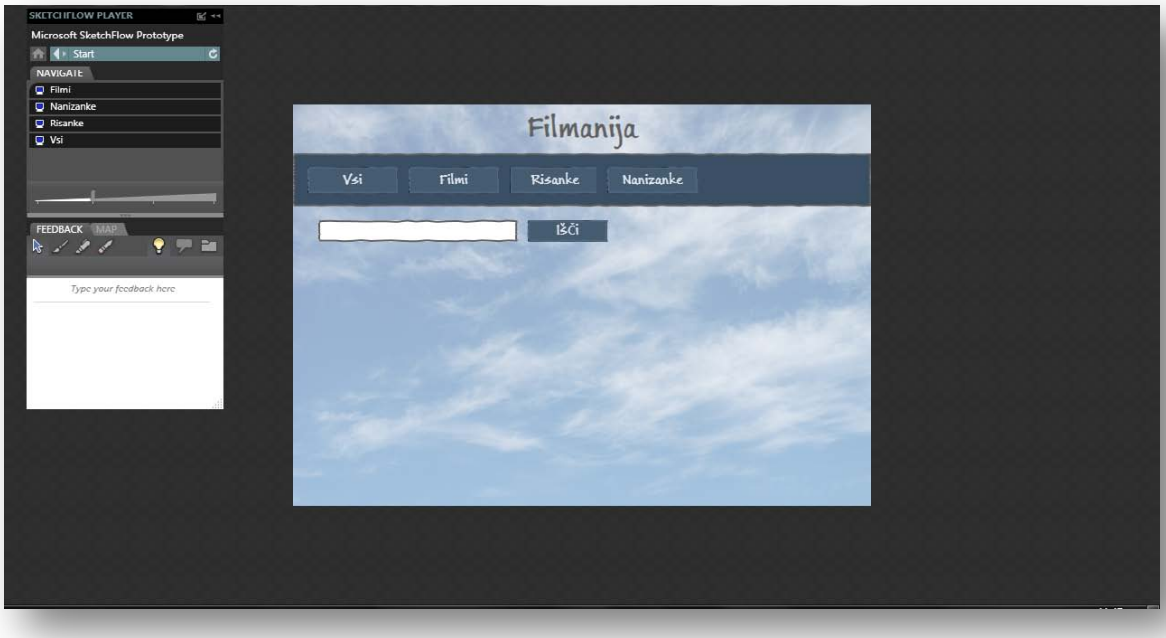


Slika 37: Prikazano okno za izvoz projekta v word dokument.

Imamo tudi, da uporabimo vnaprej pripravljen predloge. Datoteka vsebuje kazalo vsebine, kazalo slik, vse slike oken in komponent, ki jih imamo v projektu. To datoteko lahko tudi pošljemo stranki v pregled oz. lahko uporabimo tudi kot nekakšen osnutek za dokumentacijo.

4.4.11. Povratne informacije (Feedback)

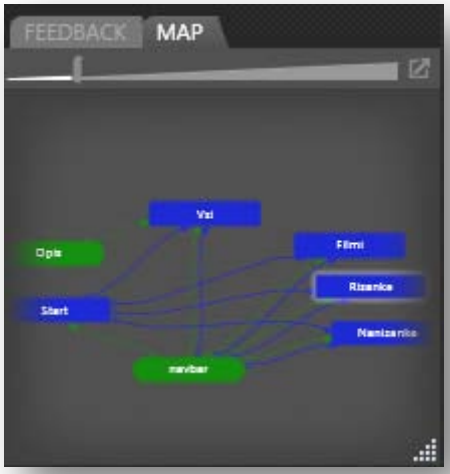
Ko stranka dobi stisnjeno datoteko projekta, jo razpakira in klikne na TestPage.html datoteko. Odpre se ji SketchFlow predvajalnik, kjer je naš projekt.



Slika 38: SketchFlow player.

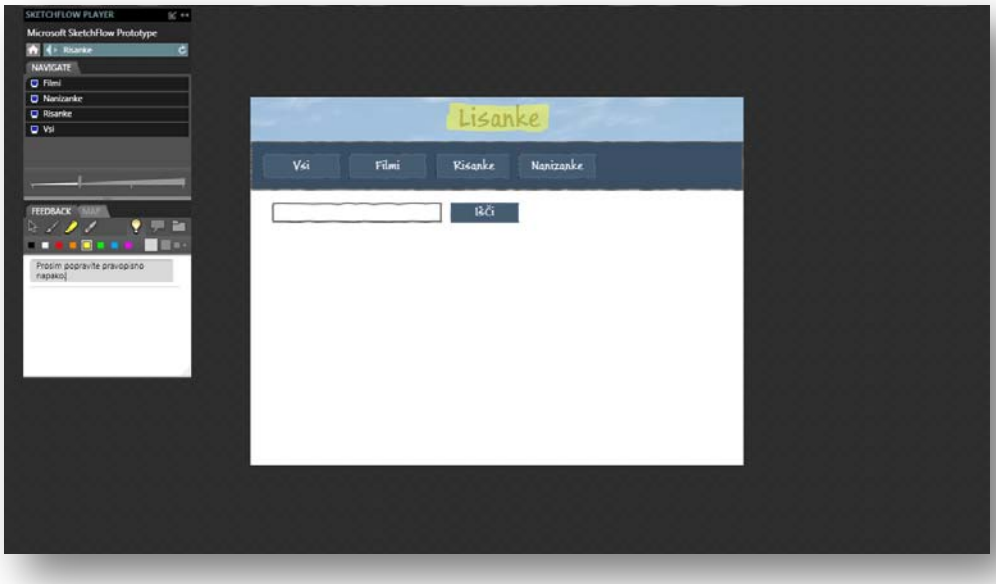
Stranka si ogleda projekt in ko opazi kakšno napako oziroma pomanjkljivost, lahko na enostaven način to tudi prikaže na SkecthFlow predvajalniku.

V našem primeru gre stranka na zavihek Map in opazi pravopisno napako, kjer namesto naslova »Risanke« piše »Lisanke«.



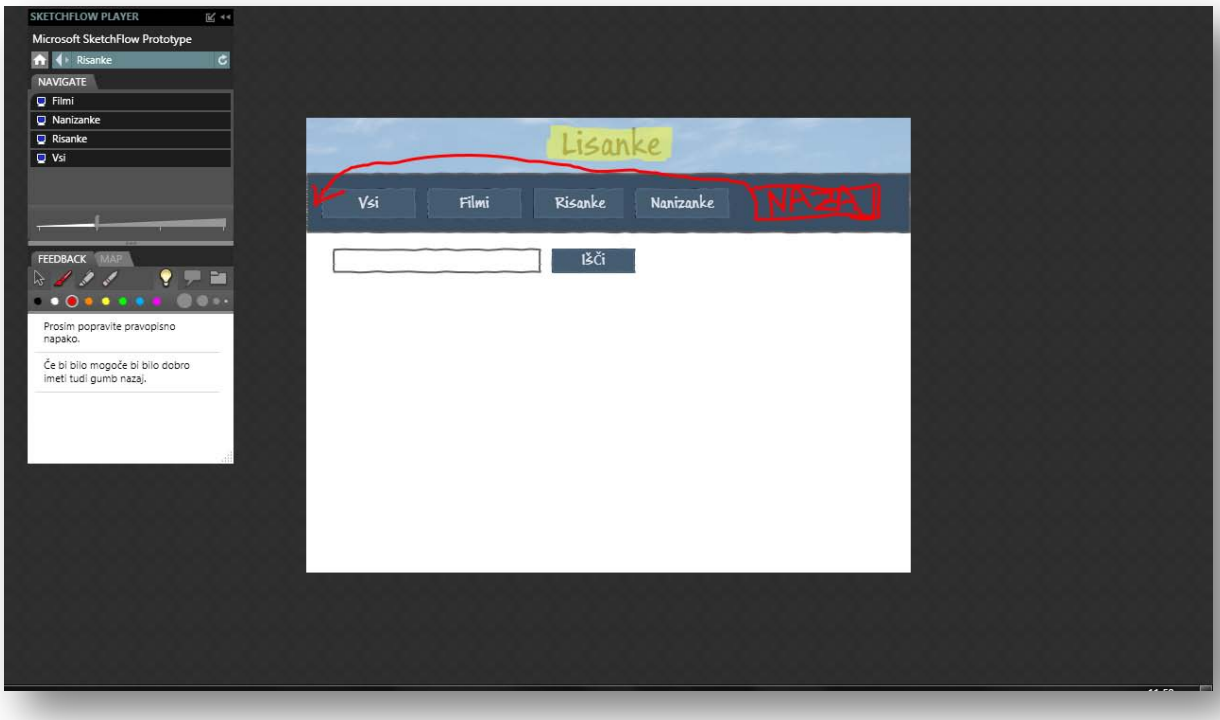
Slika 39: SkectFlow player map.

Stranka se pomakne na zavihek »Feedback«, kjer izbere ikono pisalo označevalnik in označi besedilo »Lisanke« in napiše opombo pod zavihkom »Feedback«, kot je prikazano na spodnji sliki.



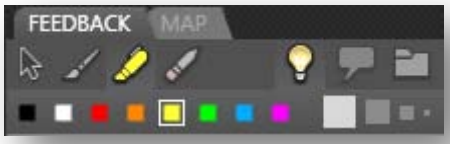
Slika 40: Prikaz napake 1.

Stranka je opazila še eno pomanjkljivost. Predlaga še možnost dodatnega gumba, s katerim bi se lahko pomikali nazaj. Izbere čopič za risanje in nariše gumb. S puščico prikaže, kje naj se ta gumb nahaja.



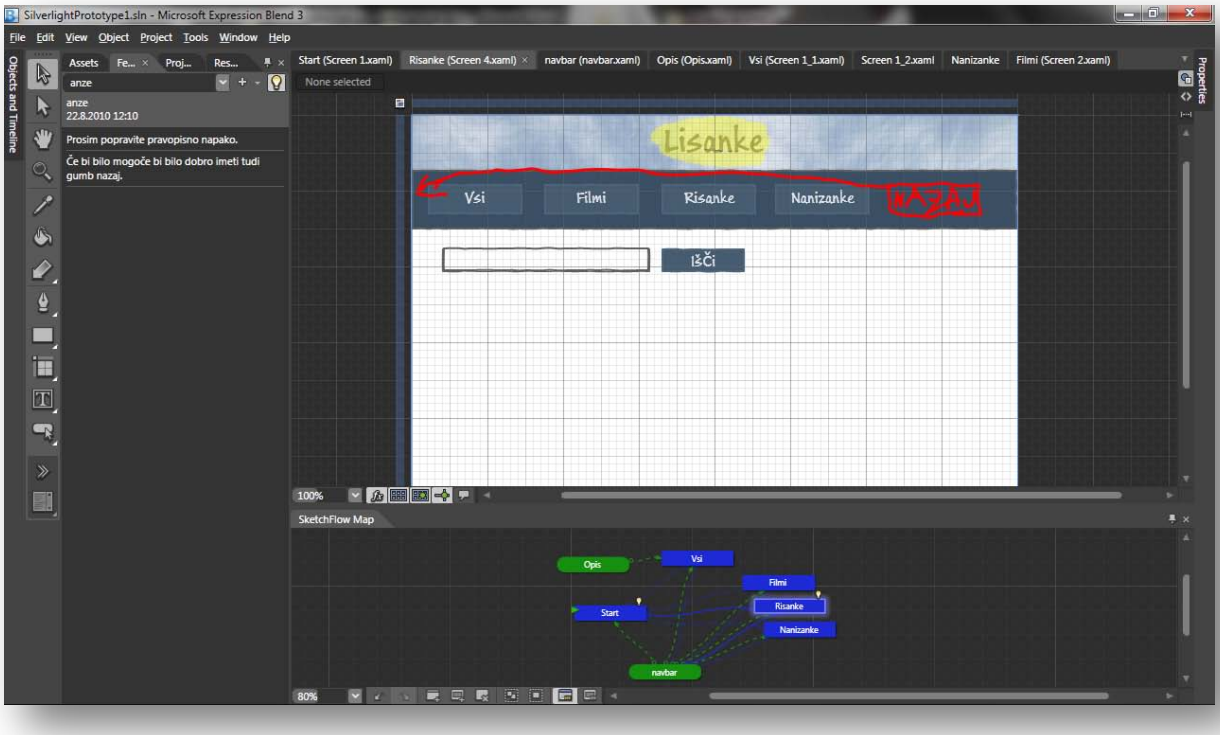
Slika 41: Prikaz napake 2.

V primeru, da stranka želi pobrisati, kar je naredila, ima na voljo tudi radirko za brisanje. Na voljo ima še navidezno skrivanje vsega, kar je dopisala. Klikne na svetilko in vsi popravki se skrijejo.



Slika 42: Feedback orodja.

Ko je stranka zadovoljna s popravki, lahko projekt enostavno izvozi. Klikne na ikono mape, ki je prikazana na zgornji sliki (desna ikona). Ponudi se ji izvoz feedback-a ali pa resetiranje. V našem primeru bo stranka izvažala. Ko izbere izvoz, se prikaže okno, kjer avtor napiše povratne informacije ter svoje ime oziroma kratico. Na podlagi tega bo programer oziroma vodja projekta vedel, koga lahko kontaktira za bolj podrobne informacije. Ko vnesemo ime avtorja, izberemo še ime datoteke in pot, kamor bomo datoteko shranili. Shrani se datoteka s končnico feedback. Ta datoteka je zelo majhna. Gre le za eno plast, ki jo lahko uvozimo v projekt in jo na enostaven prikažemo oziroma skrijemo. Stranka nam zdaj pošlje to datoteko. Ko jo dobimo, jo to uvozimo v projekt. To naredimo tako, da gremo na Window → »Feedback«. Odpre se nam zavihek »Feedback«. Kliknemo na plus znak in poiščemo datoteko s končnico feedback. V zavihku »Feedback« se nam prikažejo vse opombe, avtor in čas. Na oknih, kjer je stranka naredila popravke, se nam popravki prikažejo tudi vizualno. V »SketchFlow map« vidimo tudi, na katerih oknih so bile narejene spremembe oziroma opombe. To vidimo po ikoni v obliki žarnice. Gre le za dodatno plast, ki jo lahko poljubno skrijemo ali prikazujemo s klikom na ikono žarnice v zavihku »Feedback«.



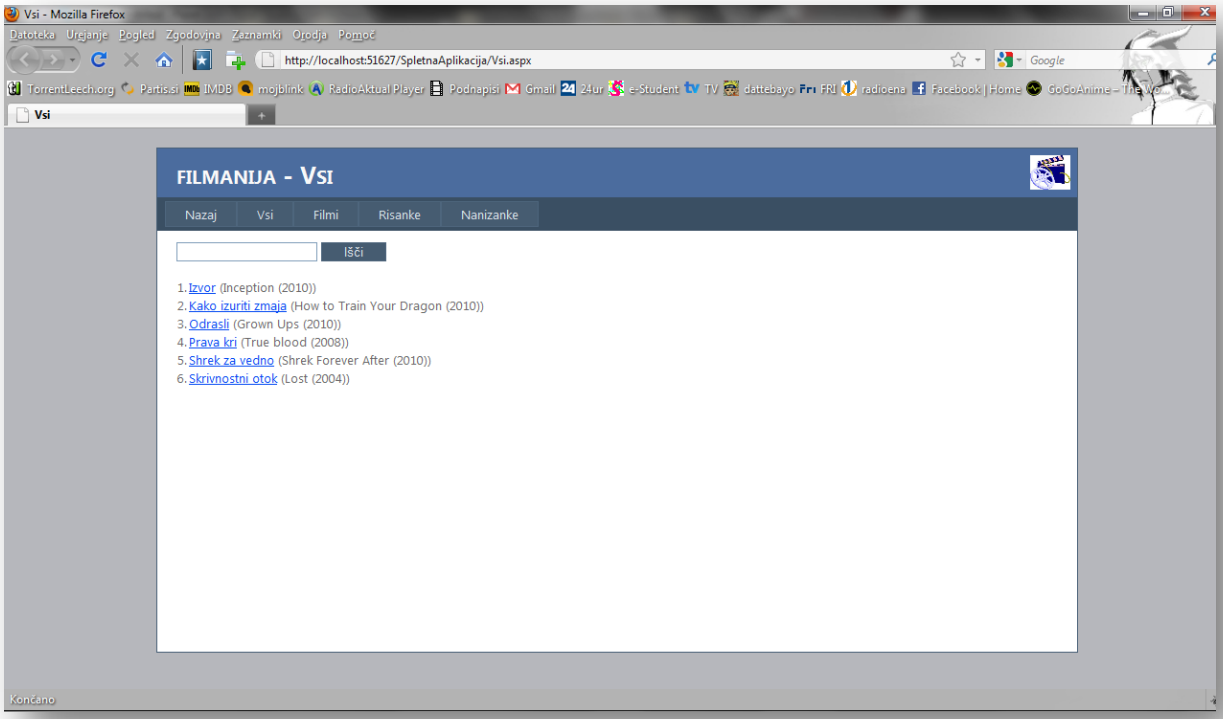
Slika 43: Uvoženi feedback.

Če zaženemo projekt, vidimo popravke tudi v predvajalniku. Dodamo lahko svoj »feedback« in ga po istem postopku, kot je opisano zgoraj, pošljemo stranki nazaj.

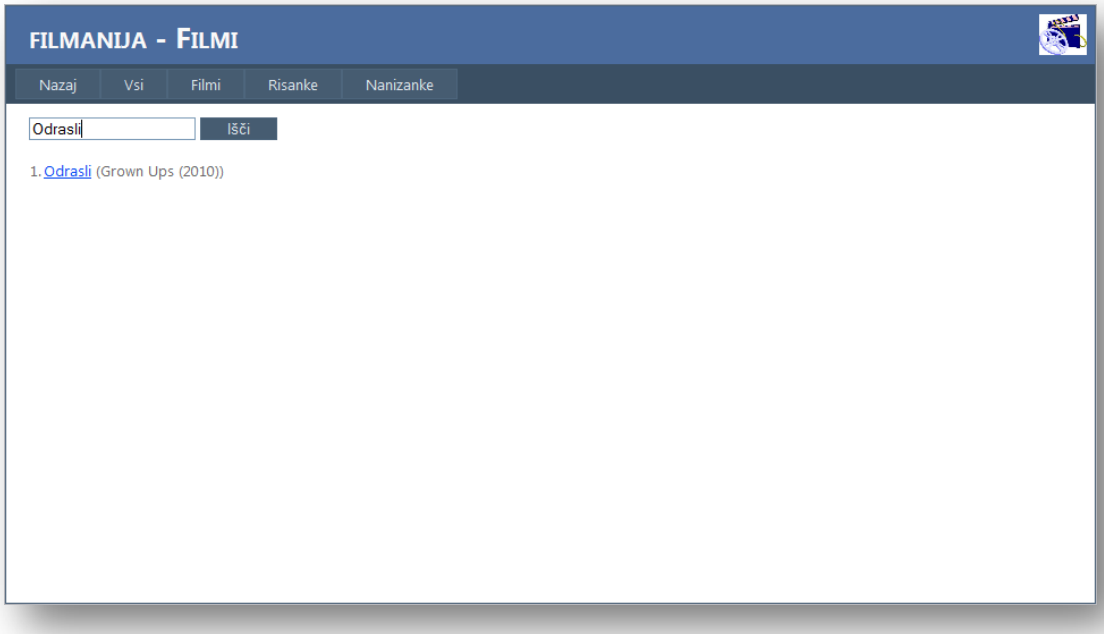
4.5. Preprosta spletna aplikacija

Ko je prototip dokončan, sledi izdelava dejanske aplikacije. Za izdelavo spletne aplikacije smo si izbrali orodje Microsoft Visual Studio 2010. Spletna aplikacija bo narejena s pomočjo ASP.NET in programskega jezika C#. Program bo vseboval tudi podatkovno bazo. Za podatkovno bazo smo izbrali Microsoft SQL Server 2008.

4.5.1. Slike končne aplikacije



Slika 44: Spletna aplikacija - osnovna stran (seznam filmov).



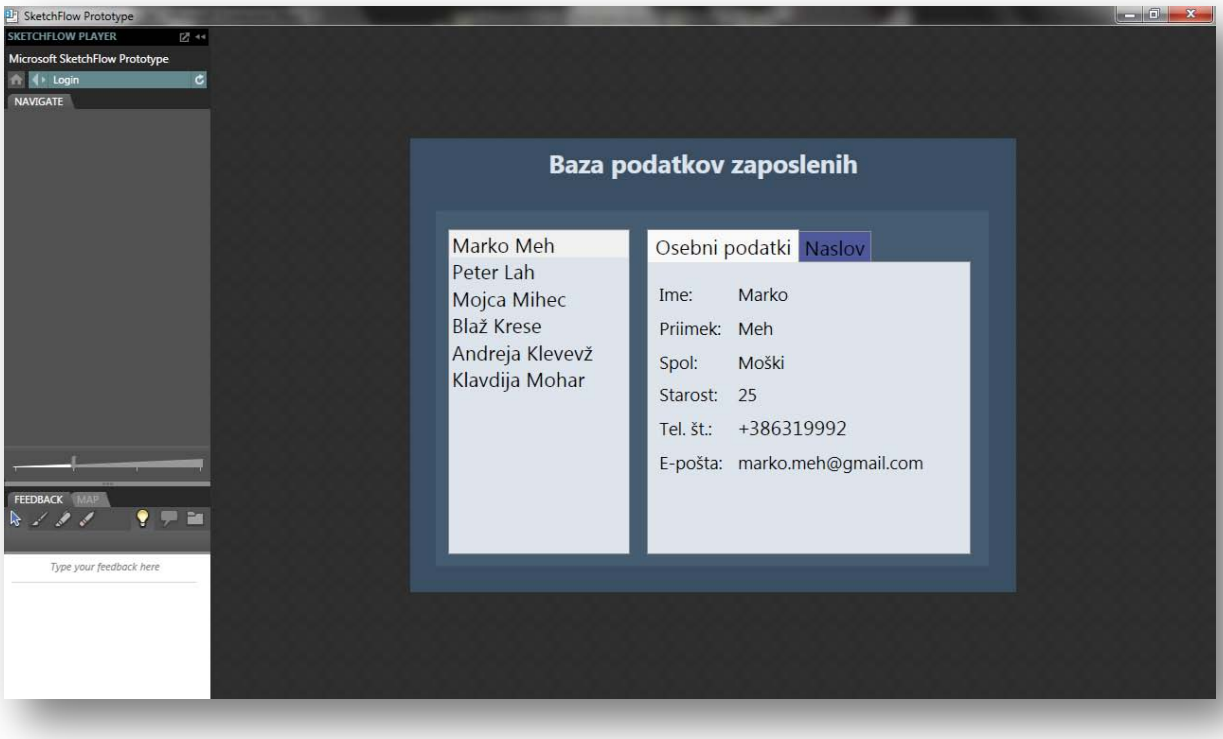
Slika 45: Iskanje filma v zavihku Filmi.

Slika 46: Opis filma Odrasli.

Slika 47: Prikaz trailerja filma Odrasli.

4.6. Preprosta WPF prototipna namizna aplikacija

Za izdelavo preprostega prototipa namizne aplikacije si bomo izbrali aplikacijo, kjer bomo imeli na voljo bazo podatkov o zaposlenih ter podrobne podatke. Za izdelavo tega prototipa si bomo pomagali z orodjem Expression Blend 3. Prototip bo temeljil na WPF platformi. Pri izdelavi aplikacije bomo čim bolj usmerjeni k temu, da bomo kasneje lahko ta prototip uporabili za kasnejše razvijanje aplikacije. Zato bomo uporabili kontrole, ki vizualno niso kot skice, temveč takšne kontrole, ki bodo že imele končni izgled. Izgled prototipa lahko vidimo na spodnji sliki.



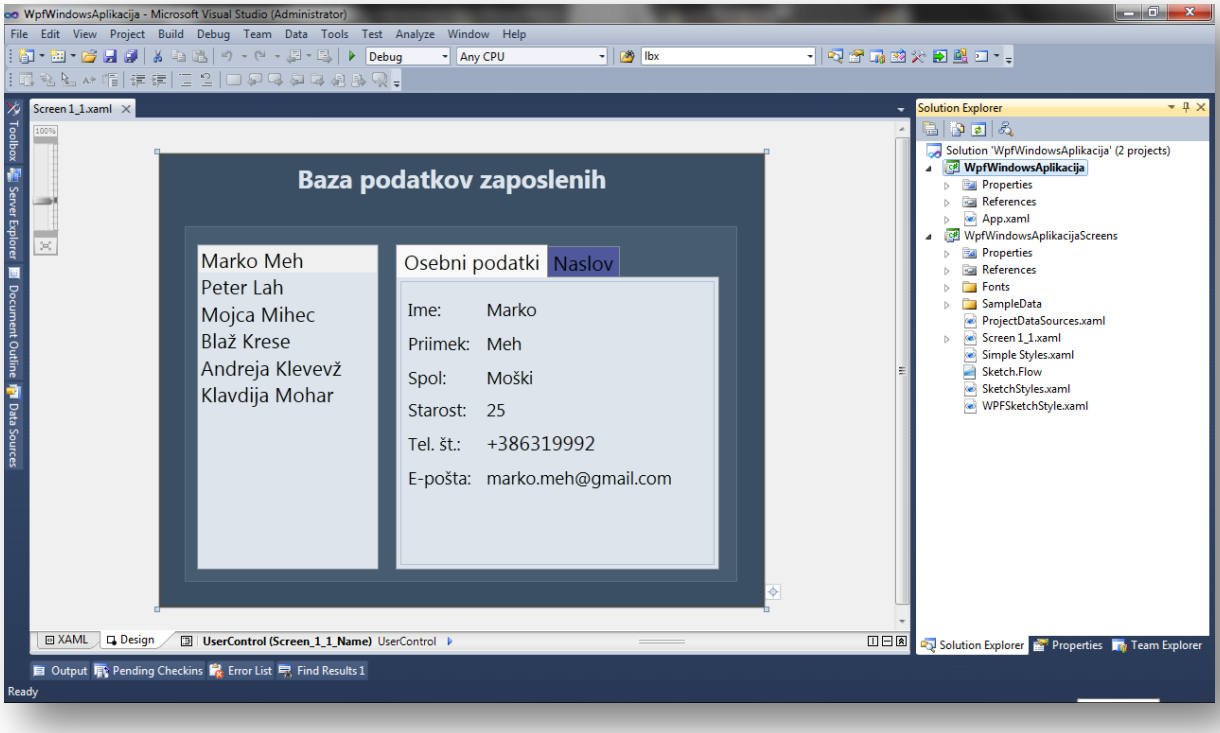
Slika 48: WPF prototip in aplikacija pred pretvorbo.

4.7. Pretvorba WPF prototipa v WPF aplikacijo.

Pretvorba WPF prototipa v WPF aplikacijo je namenjena temu, da ta prototip pretvorimo v WPF aplikacijo, ko je stranka zadovoljna z njegovim izgledom. Po pretvorbi dodamo še dodatne funkcionalnosti, ki se jih ni dalo realizirati v Expression Blendu in tako dobimo končno različico aplikacije. Niti Microsoft Expression Blend 3, niti Microsoft Expression Blend 4 trenutno ne omogočata enostavne pretvorbe prototipov. Pretvorba se pri obeh orodjih izvede na enak način. Prikaz pretvorbe WPF prototipa v WPF aplikacijo je prikazan v nadaljevanju.

4.7.1. Prikaz pretvorbe WPF prototipa v WPF aplikacijo

1. Še preden se lotimo pretvorbe, je pomembno, da naredimo varnostno kopijo aplikacije. Nato projekt prototipa, ki smo ga razvili s pomočjo Expression Blend-a, odpremo s pomočjo Microsoft Visual Studia 2010.



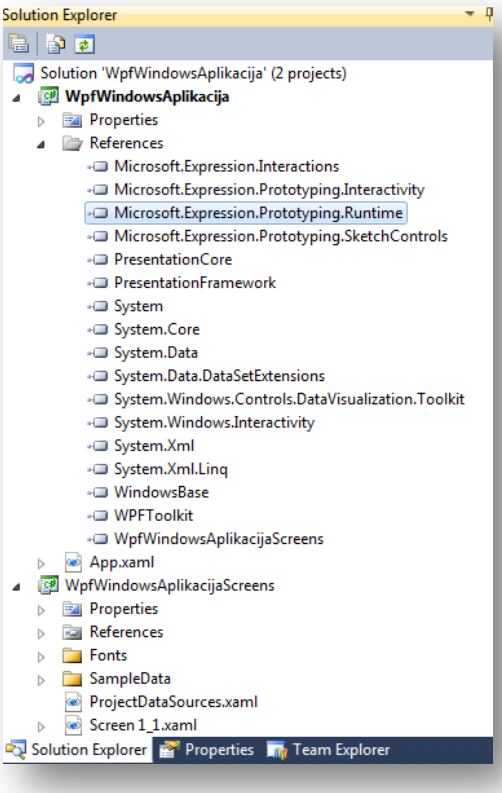
Slika 49: Microsoft Visual Studio 2010.

2. V Solution Explorer zavihku z desnim klikom kliknemo na »WpfWindowsAplikacija« in izberemo »Open Folder in Windows Explorer«. Odpre se nam raziskovalec, kjer imamo seznam datotek, ki jih vsebuje projekt.
3. V raziskovalcu poiščemo datoteko s končnico ».csproj« (WpfWindowsAplikacija.csproj) in odpremo to datoteko s programom, namenjenim oblikovanju besedila. Desni klik na datoteko in izberemo »Open with« oz. »Odpri z« in poiščemo program »Notepad«.
4. V odprti datoteki poiščemo in izbrišemo vrstice, ki so prikazane na spodnji sliki.

```
<ExpressionBlendPrototypingEnabled>false</ExpressionBlendPrototypingEnabled>
<ExpressionBlendPrototypeHarness>true</ExpressionBlendPrototypeHarness>
```

Slika 50: Koda za brisanje 1.

5. Shranimo in zapremo datoteko.
6. V Solution Explorer-ju poiščemo mapo »References«. V njej poiščemo »Microsoft.Expression.Prototyping.Runtime.dll« in ga izbrišemo.



Slika 51: Solution Explorer.

7. V Solution Explorer zavihku z desnim klikom kliknemo na »WpfWindowsAplikacijaScreens« in izberemo »Open Folder in Windows Explorer«. Odpre se nam raziskovalec, kjer imamo seznam datotek, ki jih vsebuje projekt.
8. V raziskovalcu poiščemo datoteko s končnico ».csproj« (WpfWindowsAplikacijaScreens.csproj) in odpremo to datoteko s programom, ki je namenjen oblikovanju besedila. Desni klik na datoteko in izberemo »Open with« oz. »Odpri z« in poiščemo program »Notepad«.
9. V odprti datoteki poiščemo in izbrišemo vrstice, ki so prikazane na spodnji sliki.

```
<ExpressionBlendPrototypingEnabled>false</ExpressionBlendPrototypingEnabled>
<ExpressionBlendPrototypeHarness>true</ExpressionBlendPrototypeHarness>
```

Slika 52: Koda za brisanje 2.

10. Shranimo in zapremo datoteko.
11. V Solution Explorer-ju poiščemo mapo »References«. V njej poiščemo »Microsoft.Expression.Prototyping.Runtime.dll« in ga izbrišemo.
12. V Solution Explorer poiščemo »App.xaml« in ga razširimo. Odpremo »App.xaml.cs«.
13. V »App.xaml.cs« poiščemo kodo, ki je prikazana na spodnji sliki.

```
[assembly: Microsoft.Expression.Prototyping.Services.SketchFlowLibraries("WpfWindowsAplikacija.Screens")]
```

Slika 53: Koda za brisanje 3

Ko najdemo to vrstico kode, si zapomnimo »WpfWindowsAplikacij.Screens« za kasnejšo rabo, nato izbrišemo to vrstico kode.

14. V »App.xaml.cs« poiščemo kodo, ki je prikazana na spodnji sliki.

```
this.StartupUri = new Uri(@"pack://application:,,,/Microsoft.Expression.Prototyping.Runtime;Component/WPF/Workspace/PlayerWindow.xaml");
```

Slika 54: Koda za brisanje 4.

Kodo, ki je prikazana na zgornji sliki, zamenjamo s kodo, ki je prikazana na spodnji sliki.

```
this.StartupUri = new Uri(@"pack://application:,,,/WpfWindowsAplikacija.Screens;Component/Screen 1_1.xaml");
```

Slika 55: Nova koda.

Ime »Screen1_1.xaml« je ime strani, ki se bo prikazala, ko bomo zagnali projekt.

15. Projekt zaženemo in odpre se nam WPF aplikacija (uspešno smo odstranili SketchFlow predvajalnik). [18]

Slika 56: WPF aplikacija.

5. Zaključek

Kakovost in čim krajši čas izdelave projekta sta bistvenega pomena tako za stranko kot za podjetje, ki se ukvarja z informacijskimi storitvami. Z izdelavo kakovostnih programov v najhitrejšem možnem času si podjetje širi dober glas in pridobiva stranke. To je še posebej pomembno v današnjem času recesije, ko stranke še toliko bolj pretehtajo denar in čas, ki ga bodo vložile v projekt. V teh primerih so orodja, kot je Microsoft Expression Blend 3, zelo dobrodošla, saj podjetju olajšajo delo in skrajšajo čas izdelave projekta Microsoft Expression Blend 3 omogoča oblikovalcem gradnjo privlačnih namiznih WPF ali spletnih Silverlight aplikacij in prototipov. Je vizualno in oblikovalcu prijazno orodje, ki omogoča risanje, animiranje, povezovanje z bazo podatkov, uvažanje slik, videov in 3D.

Za učinkovito izrabo zmožnosti tega orodja je sicer potrebno vložiti precej časa v učenje (povprečno 500 ur za študenta FRI), saj trenutno še ni na voljo veliko virov, ki bi se jih dalo uporabiti. Če pa želimo program uporabiti le za hitro predstavitev nekega prototipa, pa je čas učenja zelo majhen (povprečno 60 ur za študenta FRI). Preproste prototipe je namreč mogoče predstaviti v orodju le s klikanjem brez kodiranja.

V diplomski nalogi sem večji del predstavitve namenil spletni aplikaciji, saj tudi današnje tehnologije v vse večji meri težijo k spletnim aplikacijam v primerjavi z namiznimi. Dober primer predstavlja recimo prihajajoči Googlov operacijski sistem Google Chrome OS, ki bo spletni operacijski sistem. Za delovanje ne bo potreboval zmogljivega računalnika, pomnilnik bo deloval le kot trenutni pomnilnik. Vse se bo v bistvu odvijalo preko spleta.

Pri izdelavi prototipa sem imel težave pri nekaterih osnovnih stvareh, ki bi jih drugje rešil na enostaven način. Na drugi strani pa je bilo nekatere naloge tukaj lahko opraviti, medtem ko bi jih v drugih orodjih bilo težko izvesti. Res pa je, da sem se s tem orodjem prvič srečal pri pisanju diplomske naloge. V tem programu bi bilo dobro dopolniti možnost pretvorbe prototipa v dejansko aplikacijo s pomočjo čarovnika. Sedaj je potrebno spreminjati kodo, kar ni uporabniku prijazno.

Aplikacijo priporočam za uporabo, saj vedno znova prihajajo nove različice, ki vedno bolj poenostavljajo delo. S kontrolami, ki so na voljo, se da narediti marsikaj, potrebno je le znanje in veliko domišljije. Na Microsoftovih predavanjih, kjer so predstavljali beta verzijo Microsoft Expression Blend 4, so prikazali primer uporabe, kako je mogoče iz preprostega ListBox-a narediti skoraj nemogoče. ListBox je bil prikazan v obliki oblakov na modrem nebu, ki so bili različnih velikosti. Ti oblaki so se premikali in s klikom na njih je začel padati dež. Po določenem času pa so oblaki izginili. Posebej bi želel še poudariti, da je v Expression Blend-u edina omejitev znanje in domišljija. Vse je mogoče. Potreben je le pogum, da začnete z učenjem tega izjemnega orodja.

Kazalo slik

SLIKA 1: SHEMA VIZIJE 6
SLIKA 2: SHEMA PLANIRANJA 11
SLIKA 3: MODELI NAČRTOVANJA 12
SLIKA 4: TRIJE NIVOJI STORITEV 18
SLIKA 5: SHEMA RAZVOJA 23
SLIKA 6: RAZLIČICE PRODUKTA 25
SLIKA 7: SHEMA STABILIZACIJE 28
SLIKA 8: EXPRESSION BLEND. 33
SLIKA 9: DELOVNO OKOLJE PRED POPRAVKI 35
SLIKA 10: DELOVNO OKOLJE PO POPRAVKIH. 35
SLIKA 11: ZAVIHEK OBDELOVALNEGA OKNA 36
SLIKA 12: ZAVIHEK SKETCHFLOW MAP. 36
SLIKA 13: ZAVIHEK PROJECTS. 37
SLIKA 14: ZAVIHEK ASSETS. 37
SLIKA 15: ZAVIHEK OBJECTS AND TIMELINE. 38
SLIKA 16: ZAVIHEK PROPERTIES. 38
SLIKA 17: ZAVIHEK SKETCHFLOW ANIMATION. 38
SLIKA 18: PREDVAJALNIK SKETCHFLOW PLAYER. 39
SLIKA 19: NAVIGACIJSKO OKNO V PREDVAJALNIKU SKETCHFLOW PLAYER. 39
SLIKA 20: FEEDBACK V SKETCHFLOW PLAYERJU. 40
SLIKA 21: SKETCHFLOW MAP IN NODI. 40
SLIKA 22: KONTROLE, KI SO TRENUTNO NA IZBRANEM OKNU. 41
SLIKA 23: OKNO FILMANIJA. 41
SLIKA 24: IKONA KOMPONETNEGA OKNA NAVBAR. 42
SLIKA 25: KOMPONETNO OKNO NAVBAR NA OKNU FILMANIJA. 42
SLIKA 26: PRIKAZNO OKNO ZA IZBIRO XML PODATKOV. 43
SLIKA 27: PRIKAZNO OKNO ZA IZBIRO TOČNO DOLOČENIH PODATKOV. 43
SLIKA 28: OKNO OPIS, KI PRIKAZUJE OPIS FILMA. 44
SLIKA 29: KONTROLE, KI BOJO UPORABLJENE ZA ANIMACIJO. 44
SLIKA 30: KONTROLE, KI SO TRENUTNO NA IZBRANEM OKNU. 45
SLIKA 31: PRVA STVAR, KI JO NAREDIMO PRI ANIMACIJI. 45
SLIKA 32: PRIKAZ SLIKE PO KLIKU NA IKONO PLUSA. 45
SLIKA 33: PRIKAZ PREMIKA PRVEGA DELA ANIMACIJE. 46
SLIKA 34: PRIKAZ REZULTATA ISKANJA. 46
SLIKA 35: PRIKAZ OPISA FILMA. 47
SLIKA 36: PRIKAZ TRAILERJA. 47
SLIKA 37: PRIKAZANO OKNO ZA IZVOZ PROJEKTA V WORD DOKUMENT. 49
SLIKA 38: SKETCHFLOW PLAYER. 49
SLIKA 39: SKECTFLOW PLAYER MAP. 50
SLIKA 40: PRIKAZ NAPAKE 1. 50
SLIKA 41: PRIKAZ NAPAKE 2. 51
SLIKA 42: FEEDBACK ORODJA. 51
SLIKA 43: UVOŽENI FEEDBACK 52
SLIKA 44: SPLETNA APLIKACIJA - OSNOVNA STRAN (SEZNAM FILMOV). 53
SLIKA 45: ISKANJE FILMA V ZAVIHKU FILMI. 54
SLIKA 46: OPIS FILMA ODRASLI. 54
SLIKA 47: PRIKAZ TRAILERJA FILMA ODRASLI. 55
SLIKA 48: WPF PROTOTIP IN APLIKACIJA PRED PRETVORBO. 56

SLIKA 49: MICROSOFT VISUAL STUDIO 2010. 57
SLIKA 50: KODA ZA BRISANJE 1. 57
SLIKA 51: SOLUTION EXPLORER. 58
SLIKA 52: KODA ZA BRISANJE 2. 58
SLIKA 55: NOVA KODA. 59
SLIKA 56: WPF APLIKACIJA. 59
SLIKA 53: KODA ZA BRISANJE 3. 59
SLIKA 54: KODA ZA BRISANJE 4. 59

Kazalo tabel

TABELA 1: TIPIČNE ZADOLŽITVE VLOG V FAZI VIZIJE 7
TABELA 2: TIPIČNE ZADOLŽITVE VLOG V FAZI PLANIRANJA 11
TABELA 3: MODELI NAČRTOVANJA 12
TABELA 4: FUNKCIONALNE SPECIFIKACIJE 20
TABELA 5: VLOGE IN NJIHOVE ZADOLŽITVE 21
TABELA 6: VSEBINA PROJEKTNEGA PLANA 22
TABELA 7: TIPIČNE ZADOLŽITVE V FAZI RAZVOJA 23
TABELA 8: TIPIČNE ZADOLŽITVE VLOG V FAZI STABILIZACIJE 29

Viri in literatura

[1] Microsoft Corporation, Analyzing Requirements and Defining Microsoft® .NET Solution Architectures, Microsoft Press, 2003

[2] Scott Wilson, Bruce Maples, Tim Landgrave, Analyzing Requirements and Defining Solution Architectures, Kizan Corporation, 1999

[3] (2010) Osnove projektnega vodenja. Dostopno na: http://www.ra-sinergija.si/projektno_vodenje/1__uvod.html

[4] (2010) Wikipedia - WPF, Silverlight in XAML. Dostopno na: http://sl.wikipedia.org/wiki/Windows_Presentation_Fundation_vs._Windows_Forms

[5] (2010) Wikipedia - Kaj je projekt. Dostopno na: http://sl.wikipedia.org/wiki/Projekt

[6] (2002) MSF Process Model v. 3.1. Dostopno na: http://download.microsoft.com/download/7/7/7/777104c9-506e-47c9-9da4-9e23138be493/MSF%20Process%20Model%20v.%203.1.pdf

[7] (2006) MSF Process model cookbook. Dostopno na: http://developers.de/media/p/436/download.aspx

[8] (2010) Microsoft Visio. Dostopno na: http://office.microsoft.com/sl-si/visio/

[9] (2009) Alternative Microsoft Visio. Dostopno na: http://www.osalt.com/visio

[10] (2010) Opis Microsoft Expression Blend. Dostopno na: http://www.microsoft.com/slovenija/msdn/expression/expression-blend.mspx

[11] (2010) Opis Microsoft Project. Dostopno na: http://www.b2.eu/racunalniski-tecaji/projektno-vodenje.aspx

[12] (2009) Alternative Microsoft Project. Dostopno na: http://www.osalt.com/project

[13] (2010) Opis Microsoft Visual Studio. Dostopno na: http://www.reproms.si/prodaja/razvojna_orodja.wlgt

[14] (2009) Alternative Expression Blend. Dostopno na: http://www.osalt.com/visual-studio

[15] (2009) Nekaj o Microsoft Expression Blend 3. Dostopno na: http://www.msblogs.si/post/Primeri-uporabe-Microsoft-Expression-Blend-3.aspx

[16] (2010) Opis Microsoft Expression Blend 4. Dostopno na: http://www.microsoft.com/expression/products/Blend_Features.aspx

[17] (2010) Video primeri Expression Blend 3. Dostopno na: http://expression.microsoft.com/en-us/cc197141.aspx

[18] (2010) Kako pretvoriti WPF aplikacijo z uporabo Visual C#. Dostopno na: http://msdn.microsoft.com/en-us/library/ee371158%28Expression.30%29.aspx