UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,  
RAČUNALNIŠTVO IN INFORMATIKO

Filip Horvat

**Feature driven development**

Raziskovalno delo

Maribor, januar 2021

**KAZALO**

1 Uvod 1

2 Feature driven development 2

2.1 Motivacija 2

2.2 Kaj je *FDD* 2

2.3 Člani ekipe *FDD* 3

2.4 5 korakov v *FDD* življenjskem ciklu projekta 3

2.4.1 Razvij celoten model (angl. *Develop the overall model*) 3

2.4.2 Sestavljanje seznama funkcionalnosti (angl. *Build feature list*) 4

2.4.3 Načrt glede na funkcionalnost (ang. *Plan by feature*) 4

2.4.4 Oblikovanje glede na funkcionalnost (angl. *Desing by feature*) 4

2.4.5 Zgraditi glede na funkcionalnost (ang. Build by feature) 4

3 Uporaba *FDD* na projektu *What to watch* 5

3.1 Kratka predstavitev projekta 5

3.2 Vodenje projekta s modelom *FDD* 5

seznam virov 7

# Uvod

V današnjem svetu se vsakodnevno srečamo z računalniki in programsko opremo, ki na ga poganja. Zaradi tega je povpraševanje po programski opremi vse večje. Razvoj te, mora zato biti hiter in zagotavljati ustrezno delovanje. Hkrati mora biti programska odporna na napake in nuditi določeno stopnjo varnosti.

Da izpolnimo te zahteve se moramo dobro pripraviti in se držati določenih smernic. Ta lahko zagotovimo z izbiro ustrezne metode razvoja programske opreme. Teh metod je veliko, in se nenehno spreminjajo.

Razvoj bi naj potekal po sedmih stopnjah življenjskega cikla razvoja programske opreme. Zavedati se moramo, da se razvoj nikoli ne ustavi. Vedno so prisotne napake, pomanjkljivosti in možne izboljšave. Zaradi tega je pametno izbrani čim boj dinamični način razvoja, ki pa v ospredje postavi samo programsko opremo. Torej, da bo ta vsebovala določene vnaprej določene funkcionalnosti, ki jih definira stranka oz. uporabniki.

Pri razvoju programske opreme moramo paziti, da ne zapademo v razmišljanje več je boljše. V tem primeru lahko naš produkt postane nestabilen, nedefiniran in brez natančne vizije. Da se temu izognemo, si lahko za metodo razvoja programske opreme izberemo razvoj z poudarkom na funkcionalnostih oz. angl. *feature driven development*.

# Feature driven development

## Motivacija

Ko govorimo o agilnih metodah razvoja programske opreme, najprej pomislimo na *scrum* [1]. Gre za najpopularnejšo metodologijo razvojnih ekip, zadnjih 13 let. Zraven *scuma* pa se razvijajo in uporabljajo nove agilne metode, ki so lahko za določene projekte primernejše, kot je *Feature driven development* (v nadaljevanju *FDD*). Gre za neke vrste hibrid *scrum*-a in ostalih modernih agilnih metodologij. *FDD* je primerna izbira za dolgotrajne projekte, na katerih dela večje število velikih razvojnih skupin [2,3].

## Kaj je *FDD*

*FDD* je leta 1997 razvil Jeff De Luca, med delom na več mesečnem projektu v Singapurju, na katerem je delalo več kot 50 ljudi. Da bi pomagal razvojnim ekipam, postati bolj adaptivne in odzivne, na nove zahteve stranke, je zasnoval razvojni model, ki ga sestavlja pet korakov. Ti koraki se osredotočijo na razvoj novih funkcionalnosti programske opreme v kratkih iteracijah. Koraki zagotavljajo, da projekt raste in da se novi razvijalci lažje vključijo v proces razvoja. Iteracije trajajo po en do dva tedna, kjer razvijalci sledijo naboru dobrih inženirskih praks, da razvijejo majhen nabor določenih funkcionalnosti.[2,3,4]

Nekaj dobrih inženirskih praks [2]:

* ***Domain object modeling:*** Ekipa sestavi razredni diagram, da opiše objekte v domeni in njihova razmerja.
* ***Developong by feature:*** Če funkcionalnost ne more biti implementirana v največ dveh tednih (dolžina iteracije), jo je potrebno razdeliti na več manjših funkcionalnosti.
* ***Individual class (code) ownership:***vsakemu razredu oz. skupini kode je dodeljen lastnik.
* ***Feature teams:***Ker določeno funkcionalnost sestavlja več razredov oz. skupin je potrebno vodstvo in posledično sledenje dodeliti/združiti v skupino.
* ***Inspecions:*** *FDD* ekipe opravljajo inšpekcije kode, da se hitro identificira in reši potencialen problem
* ***Configuration management:*** Dokumentacija kode
* ***Progress reports:***Projektni menedžerji sproti izdelujejo poročila napredka

## Člani ekipe *FDD*

Vsaka ekipa *FDD* bi naj vsebovala naslednje vloge [2]:

* **Projektni menedžer** (angl. *project manager)****:*** Nadzoruje celoten projekt
* **Glavni arhitekt** (angl. chief arhitect): Zadolžen za načrtovanje in modeliranje sistema.
* **Razvojni menedžer** (angl. *development manager*): vodi razvojno ekipo in nadzira dnevne programerske aktivnosti
* **Glavni programer** (angl. *chief programmer*): pomaga pri analizi in načrtovanju sistema
* **Lastnik razreda** (angl. *class owner*): Član manjše razvojne skupine, ki jo vodi glavni programer. Njegove naloge so načrtovanje, programiranje, testiranje in sestavljanje dokumentacije novih funkcionalnosti
* **Strokovnjak domene** (angl. *domain expert*): Član ekipe, ki razume in zna interpretirati probleme stranke. Celoten razvoj programske opreme je odvisen od njegovega znanja v določeni domeni.

## 5 korakov v *FDD* življenjskem ciklu projekta

Razvij celoten model (angl. *Develop the overall model*)

V prvem koraku *FDD* definiramo model domene. Za programsko opremo ustvarimo model, ki bo reševal zastavljen problem. Razvojna ekipa v tem koraku tesno sodeluje z glavnim arhitektom, da definirajo obsežnost in kontekst sistema. Priporočeno je, da ekipa/ekipe ustvarijo več modelov, ki jih nato združijo v celoto oz. skupen model [2,3].

Sestavljanje seznama funkcionalnosti (angl. *Build feature list*)

Rezultat tega koraka je seznam funkcionalnosti. Ta seznam je podobne *scrum*-ovemu *product backlog*-u. Identificirati je potrebno funkcionalnosti, ki so pomembne stranki. Te funkcionalnosti so definirane, kot *akcije, rezultati* in *objekti* npr. »Validacija uporabnika ob prijavi«. Če je časovna zahtevnost posamezne funkcionalnosti večja kot dva tedna, jo razdelimo na več manjših funkcionalnosti[2,3].

Načrt glede na funkcionalnost (ang. *Plan by feature*)

Določiti je potrebne vrstni red razvoja in implementacije seznama funkcionalnosti. V tem koraku moramo identificirati potencialna tveganja, odvisnosti do ostalih funkcionalnosti in ugotoviti količino dela za posameznega programerja. Vsako definirano funkcionalnost nato razdelimo med programerje[2,3].

Oblikovanje glede na funkcionalnost (angl. *Desing by feature*)

Glavni programer v tem koraku določi katere funkcionalnosti bodo oblikovane v 2 tedenski iteraciji. Vsaki funkcionalnosti dodeli tudi prioriteto in se odloči, kdo bo na ekipi funkcionalnosti. Preden se lahko razvoj nadaljuje je potrebno dokončati tudi pregled oblikovanja[2,3].

Zgraditi glede na funkcionalnost (ang. Build by feature)

V tem koraku se dokončana in testirana koda zgradi v celoto oz. prototip. Oblikovati se mora tudi uporabniški vmesnik. Vse skupaj se nato testira kot celota. Če so vsi testi uspešni se dokončana verzija doda glavni aplikaciji in postane dostopna končnemu uporabniku[2,3].

# Uporaba *FDD* na projektu *What to watch*

## Kratka predstavitev projekta

Cilj projekta »*What to watch«* je ustvariti aplikacijo, ki uporabniku pomaga pri izbiri multimedijske vsebine, ki si bi jo rad ogledal v prostem času. Aplikacija pri izbiri pomaga z algoritmom izbire glede na predhodno ogledane vsebine in vprašalnikom. Uporabnik lahko izbiro filtrira glede na njemu dostopne servise (npr. *netflix*, *hbogo,…*). Na svojem profilu si lahko ustvari zbirke najljubših vsebin in jih deli s prijatelji. Primarna platforma aplikacije je pametni telefon.

## Vodenje projekta s modelom *FDD*

Glavna funkcionalnost aplikacije je bila izbira multimedijske vsebine. To je obsežna funkcionalnost, katere časovna zahtevnost krepko presega 2 tedna, zato smo jo razdeli.

Funkcionalnosti smo definiramo z zgodbama »Aplikacija uporabniku ponudi izbiro iz obsežne zbirke multimedijskih vsebin.«, »Aplikacija za izbiro uporabi algoritem, ki upošteva uprabnikove pretekle izbire.« Potrebovali smo izbirni algoritem in podatkovno bazo velike količine multimedijskih vsebin. Zbiranje vsebin smo dalje razdelili glede na žanro, časovno obdobje in dostopnost. Algoritem smo deli na preteklo izbiro uporabnika in na izbiro glede na ostale podobne uporabnike.

Te funkcionalnosti so sestavljaje osnovo aplikacije. Neodvisno od njihovega razvoja, smo lahko začeli z izdelavo uporabniškega vmesnika. Definirala ga je zgodba »Uporabnik do funkcionalnosti aplikacije dostopa do modernega, minimalističnega grafičnega vmesnika, ki mora biti odziven in jasen.«, katero smo razdelili na več manjših delov npr. »Za vsako vsebino naredimo stran, kjer so jasno predstavljene informacije in ocene.«.

Aplikacija je morala delovati na več različnih platformah, kot je pametni telefon *android, iOS,…*, tabličnih računalnikih in namiznih računalnikih. Za vsako različno platformo smo ustvarili svojo zgodbo: »Aplikacija mora delovati na platformi *Windows* *10*, brskalnik *Chrome*«,…

V času razvoja smo algoritem nenehno testirali in spreminjali uteži glede na pridobljene rezultate. Enako velja za uporabniški vmesnik, ki je moral biti končnemu uporabniku všeč in zagotavljati odzivno delovanje.

Ko smo zbrali večino multimedijskih vsebin, dokončali algoritem in zasnovali primeren uporabniški vmesnik, smo te funkcionalnosti združili v celoto in ustvarili prvi prototip. Prototip je bil deležen dolgega testiranja različnih potencialnih uporabnikov različnih starosti, narodnosti, spola, izobraženosti. Ko je bila večina testerje zadovoljena z aplikacijo smo jo izdali javnosti.

Sledi razvoj novih funkcionalnosti. Pomembno je, da smo v nenehnem kontaktu z dejanskimi uporabniki, da razvoj aplikacije vodi v pravo smer.

# **seznam virov**

1. Scrum.org: *What is SCRUM?* Dostopno na: <https://www.scrum.org/resources/what-is-scrum> [2. 1. 2021].
2. Lucidchart: *Why use feature driven development*. Dostopno na: [https://www.lucidchart.com/blog/why-use-feature-driven-development](https://www.lucidchart.com/blog/why-use-feature-driven-development%20) [2. 1. 2021].
3. Planview: *What is FDD in Agile?*. Dostopno na: <https://www.planview.com/resources/articles/fdd-agile/> [2. 1. 2021].
4. ProductPlan: *Feature driven development.* Dostopno na: <https://www.productplan.com/glossary/feature-driven-development/> [2. 1. 2021]
5. Agile Modeling: *Feature driven development (FDD) and Agile Modeling*. Dostopno na:

[http://agilemodeling.com/essays/fdd.htm](http://agilemodeling.com/essays/fdd.htm%20) [2. 1. 2021]