

UNIVERZITET DŽEMAL BIJEDIĆ U MOSTARU  
FAKULTET INFORMACIJSKIH TEHNOLOGIJA

Završni rad nakon I ciklusa

# Agilni razvoj softvera

Student: Ernad Husremović, DL 2792

Mentor: prof. dr. Emina Junuz

Mostar, septembar 2018.

*za ljubav mojim roditeljima*

# Sadržaj

Sažetak.....	6
1 Uvod.....	7
1.1 Manifest agilnog razvoja softvera.....	7
1.2 Osnovni principi u agilnom razvoju softvera.....	7
1.3 Usporedba tradicionalnih i agilnih metoda razvoja.....	8
1.3.1 Tradicionalne metode.....	8
1.3.2 Agilne metode.....	9
2 Ekstremno programiranje (XP).....	10
2.1 Uvod.....	10
2.2 Organizacija XP tima.....	10
2.3 Restruktuiranje - refactoring programskog kôda.....	11
2.4 Programiranje u paru.....	11
2.5 Testiranje na prvom mjestu (TDD).....	11
2.5.1 Rust TDD i refactoring primjer.....	11
3 “Scrum”.....	16
3.1 Uvod.....	16
3.2 Dnevni “Stand-up” sastanci (eng. stand-up meetings).....	16
3.3 “Sprintovi”.....	17
3.3.1 Centralna lista korisničkih priča (eng. Product backlog).....	17
3.3.2 Lista priča novog sprinta (eng. Spring backlog).....	17
4 “Lean software development”.....	18
4.1 Uvod.....	18
4.2 Eliminacija otpada.....	18
4.3 Kvalitet izrade.....	18
4.3.1 Kreiranje znanja.....	19
4.3.2 Donošenje odluka u najboljem trenutku.....	19
4.3.3 Brze isporuke.....	19
4.3.4 Uvažavanje ljudi.....	19
4.3.5 Optimiziranje cjeline.....	19
5 “Kanban”.....	20

5.1 Uvod.....	20
5.2 “Kanban” ploča.....	20
6 Zajedničke prakse i koncepti.....	23
6.1 Upravljanje revizijama izvornog kôda (VCS).....	23
6.1.1 “Github” repozitorij.....	25
6.2 Multifunkcionalni tim.....	26
6.3 Agilni trener.....	27
7 F18 - “knjigovodstvo za Bosance”.....	28
7.1 Uvod.....	28
7.1.1 Cilj projekta.....	28
7.1.2 “Ad-hoc” rješenja, dugoročne glavobolje.....	29
7.2 Preduzeće “bring.out”.....	29
7.2.1 Razvojni period, developeri.....	29
7.2.2 Kriza projekta.....	30
7.3 Korištene tehnologije.....	30
7.3.1 Harbour programski jezik.....	30
7.3.2 Pristup PostgreSQL bazi iz “harbour”-a.....	31
7.3.3 Više-platformski pristup.....	34
7.4 Upravljanje projektom.....	34
7.4.1 Poslovni kontekst projekta.....	34
7.4.2 Upotrebljena vrijednost.....	34
7.5 Git istorija projekta 2014-2018.....	35
7.6 Glavni ciljevi razvoja “F18” u periodu 2016-2018.....	36
7.6.1 Čitljivost izvornog kôda (eng. source code readability).....	36
7.6.2 Promjene na sloju pristupa podataka (eng. data layer).....	36
7.6.3 Realizacija programskih zahtjeva korisnika.....	36
7.7 Pregled F18, korisnički nivo.....	37
7.8 Postavke - parametri F18.....	37
7.9 FIN - Finansijsko poslovanje.....	38
7.10 FAKT - modul za fakturisanje.....	40
7.11 Ostali programski moduli.....	42
8 Razvoj projekta “F18” u periodu 2014-2018.....	43
8.1 Uvod.....	43

8.2 “Atom” editor i podrška za “harbour”.....	43
8.2.1 Atom “language-harbour” ekstenzija.....	44
8.2.2 Atom “linter-harbour” ekstenzija.....	45
8.2.3 Atom editor “harbour-plus” ekstenzija.....	46
8.2.4 Atom editor osnovne funkcije.....	47
8.2.5 Zaključak.....	49
8.3 Eliminacija programskog “otpada”.....	50
8.3.1 Stanje 2014.....	50
8.3.2 Stanje 2018.....	51
8.3.3 Zaključak.....	51
8.4 “F18 update” - provjera i instalacija novih verzija.....	52
8.4.1 Zaključak.....	53
8.5 F18 automatski izvještaji o greškama (eng. bug reports).....	54
8.5.1 Serverski log.....	55
8.5.2 “Call stack” i stanje sistema u trenutku greške.....	56
8.5.3 Svrha i značaj automatskih “bug report”-a.....	57
8.5.4 Zaključak.....	57
8.6 F18 automatska integracija (CI).....	58
8.6.1 Zaključak.....	60
9 Iz osobne prakse.....	61
9.1 Software koji se ne koristi je otpad.....	61
10 Zaključak.....	62

## **IZJAVA O AUTORSTVU**

Ja, **Husremović (Šekib) Ernad**, student Fakulteta informacijskih tehnologija, Univerziteta „Džemal Bijedić“ u Mostaru, pod punom moralnom, materijalnom i krivičnom odgovornošću, izjavljujem da je rad pod naslovom:

### **“Agilni razvoj softvera”**

u potpunosti rezultat sopstvenog istraživanja, gdje su korišteni sadržaji drugih autora jasno označeni pozivanjem na izvor i ne narušavaju bilo čija vlasnička ili autorska prava.

U Sarajevu, 13.09.2018.

---

Ernad Husremović, LK 28110A536

# Sažetak

Agilni razvoj softvera predstavlja pomak od striktno inžinjerske linearne i planirane paradigme prema adaptivnoj i manje formalnoj paradigmii baziranoj na ideji iskorištavanja resursa koji su na raspolaganju kako bi se klijentu isporučio softver koji radi. Može se reći da je agilni razvoj pomak od ideje da je samo savršeno dovoljno dobro ka ideji da proizvod ne mora biti savršen da bi bio dovoljno dobar i da uvijek postoji prostor za usavršavanje. U prvom dijelu rada biće prikazana osnovna ideja agilnog razvoja softvera kroz Manifest njegovih glavnih tvoraca, potom će se napraviti usporedba tradicionalnih metoda i metoda agilnog razvoja softvera, te na kraju prikaz najznačajnih metoda agilnog razvoja. U drugom dijelu rada će se prezentovati četverogodišnji ciklus razvoja realnog projekta, te analizirati realizacija u kontekstu primjene koncepata agilnog razvoja.

## ***Ključne riječi:***

softver, agilni razvoj, upravljanje projektima, ekstremno programiranje (XP), scrum, kanban, lean, psihologija radnog mjesti, motivacija, visualization, atom programmerski editor, harbour programski jezik, xBase, DBF, PostgreSQL, upravljanje revizijama izvornog kôda (VCS), git, testiranje na prvom mjestu (TDD), kontinuirana integracija (CI), bring.out, F18 knjigovodstvo

## ***Keywords:***

software, agile development, project management, extreme programming (XP), scrum, kanban, lean, psychology at work, motivation, vizualizacija, atom programming editor, harbour programming language, xBase, DBF, PostgreSQL, version control system (VCS), git, test driven development (TDD), continous integration (CI), bring.out, F18 accounting

# 1 Uvod

Razvoj softvera u gotovo svakom projektu obuhvata veliki dijapazon operacija od jednostavnih repetitivnih operacija koje nalikuju radu na tvorničkoj traci preko klasično inžinjerskih tehnika koje su slične gradnji kuće ili mosta do kreativnih procesa bliskih kreiranju umjetničkih djela. Agilni razvoj softvera temelji se na Manifestu koji utvrđuje osnovne vrijednosti i principe. Na tim temeljima su nastale različite agilne metode koje unutar sebe objedinjuju određene prakse i koncepte. Agilni razvoj prepoznaće specifičnosti razvoja softvera u odnosu na druge inžinjerske grane.

## 1.1 Manifest agilnog razvoja softvera

Manifest agilnog razvoja softvera<sup>1</sup> je proglašen grupom 17 softver developerima objavljen 2001. godine:

*“Mi otkrivamo bolje načine razvoja softvera, primjenjujemo ih i pomažemo drugima da ih primjenjuju proizvodimo softver i drugima pomažemo u njegovoj izradi. Kroz ovaj rad naučili smo cijeniti:*

***Osobe i njihove interakcije*** više od procesa i alata

***Softver koji funkcioniše*** više od obimne dokumentacije

***Saradnju sa klijentom*** više nego pregovore oko ugovora

***Prilagođavanje promjenama*** više nego striktno slijedenje plana

*Iako stavke na desnoj strani imaju svoju vrijednost, mi više cijenimo stavke na lijevoj strani.”*

## 1.2 Osnovni principi u agilnom razvoju softvera

Tvorci manifesta navode principe na kojima se bazira razvoj softvera u novoj paradigmi:

*“Slijedimo sljedeće principe:*

1. *Naš najveći prioritet je zadovoljiti klijenta kroz rane i kontinuirane isporuke funkcionalnog softvera.*
2. *Prihvatamo promjene promjene u softverskim zahtjevima, čak i u ranoj fazi razvoja. Agilni procesi usvajaju promjene koje će obezbjediti klijentu kompetativnu prednost.*
3. *Stalno isporučujemo funkcionalan softver, u intervalima od nekoliko sedmica do par mjeseci, preferirajući kraće vremenske cikluse.*
4. *Poslovni ljudi i developeri trebaju kontinuirano, na dnevnoj osnovi, raditi zajedno na projektu.*
5. *Izradu projekata povjeriti motiviranim pojedincima. Obezbeđujemo im kvalitetno okruženje i svu potrebnu podršku, imamo povjerenje u njih da će posao biti završen kako treba.*
6. *Najefikasniji način prenosa informacija potrebnih za razvoj je direktna (lice u lice) komunikacija između učesnika.*

---

<sup>1</sup> <http://agilemanifesto.org/>

7. Funkcionalan softver je primarna metrika napretka projekta.
8. Agilni procesi promoviraju održivi razvoj softvera. Investitori, developeri, korisnici trebaju moći nastaviti učešće u održavanju softvera neograničeno.
9. Kontinuirano pažnja na tehničku izvrsnost i dobar dizajn povećava agilnost.
10. Jednostavnost - sposobnost da se maksimalizira količina posla koji **nije** potrebno realizovati je od esencijalne važnosti.
11. Najbolje arhitekture, zahtjevi i dizajn kreiraju samoorganizovani timovi.
12. Tim u regularnim intervalima analizira svoje djelovanje, traži načine da bude što efektivniji, te u skladu sa tim nalazima podešava i prilagođava svoje djelovanje.”

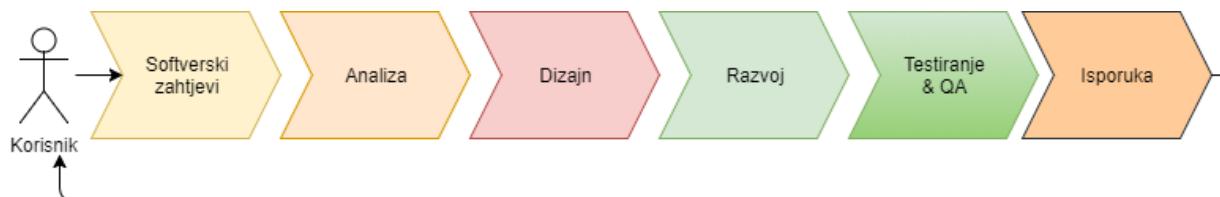
## 1.3 Usporedba tradicionalnih i agilnih metoda razvoja

### 1.3.1 Tradicionalne metode

Tradicionalne metode razvoja softvera slijede hijerarhijsku linearnu paradigmu koja odlikuje klasične inžinjerske projekte. Glavne karakteristike tradicionalnih metoda su:

- pojedinačne faze se obavljaju sekvencialno (analiza, dizajn, razvoj, testiranje, isporuka)
- pojedinačne faze (posebno u većim projektima) izvode različiti, izolovani timovi
- prije faze realizacije razrađuje se precizan i striktan vremenski plan koji je često integralni dio ugovora sa korisnikom
- Zahtjevi **definisani na početku** određuju vremenski plan, arhitekturu i realizaciju
- Obimna, ali **statična** dokumentacija
- U mnogim segmentima primjenjuju se metode koje su viđene u drugim inžinjerskim granama (npr. u građevinskim projektima)

Najpoznatiji metod koji se svrstava u tradicionalne metode razvoja softvera je “waterfall” model prikazan na Dijagramu 1.



Dijagram 1: "Waterfall" metoda u tradicionalnom razvoju softvera

Na Dijagramu 1. uočavamo dugačku povratnu petlju razvojni tim - korisnik. Korisnik u ovom modelu ima priliku da pošalje povratnu informaciju o kvaliteti produkta tek nakon što su okončane sve faze što projekta. Ova relativno kasna povratna informacija stvara velike izazove u modifikaciji početnog produkta jer korisnik gubi mogućnost da razumijeva produkt kroz sve faze, a to vodi ka tome da je njegova povratna informacija nespecifična i nefokusirana, pa često i nerazumljiva.

### **1.3.2 Agilne metode**

Bazirajući se na osnovnim vrijednostima i principima agilnog razvoja softvera, razvijeno je niz agilnih metoda koje unutar sebe obuhvataju određene elemente - prakse.

Najpoznatije metode su:

- Ekstremno programiranje (XP)
- “Scrum”
- “Lean software development”
- “Kanban”

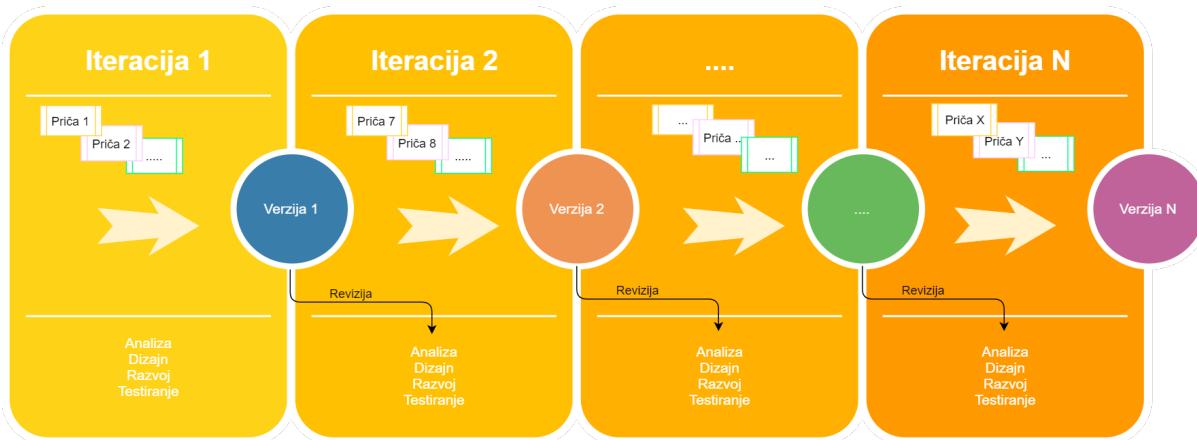
## 2 Ekstremno programiranje (XP)

### 2.1 Uvod

**Prihvatanje promjena** (eng. embracing changes) je osnovno polazište metode “Ekstremno programiranje” (XP). Dva ključna mesta unutar kojih se primjenjuje ovaj princip su granice projekta i programski kôd. Prakse i koncepti XP-a obezbeđuju da razvojni tim bude sposoban da prihvati i realizuje takve promjene na najbolji način. Zato npr. novi korisnički zahtjev ili značajne promjene izvornog kôda nisu vanredne, nego očekivane aktivnosti.

### 2.2 Organizacija XP tima

XP tim radi u sedmičnim iteracijama. Svake sedmice, tim obavlja analizu, dizajniranje, razvoj i testiranje. Osnovna jedinica rada je “priča” (eng. story). Priče predstavljaju funkcije ili dio funkcija softvera koje imaju određenu vrijednost za korisnika. Svake sedmice tim isporučuje 4-10 priča. Na kraju iteracije vrši se interna revizija softvera. Ukoliko je moguće, revizija se isporučuje krajnjim korisnicima na testiranje[AART]. Dijagram 2. prikazuje tok operacija XP tima:



Tim radi u otvorenom prostoru koji omogućava brzu izmjenu informacija i ideja putem direktnе komunikacije (lice u lice). Ako postoji mogućnost, zajedno sa timom nalaze se predstavnici korisnika i koji praktično predstavljaju produženi dio tima. Oni učestvuju u konstrukciji priča, testiranju tokom razvoja, kao i revizijama softvera na kraju iteracija. Ukoliko krajnji korisnici nisu dostupni na licu mjesta, jedan od članova tima se imenuje za zastupnika korisnika (eng. proxy user), tako da se njemu postavljaju pitanja upućena korisnicima. Najčešće proizvod-menadžer (eng. product manager) preuzima ulogu proksija.

Određene prakse su posebno istaknute u XP metodologiji:

- Restruktuiranje programskog kôda (eng. refactoring)
- Programiranje u paru (eng. pair programming)
- Testiranje na prvom mjestu (eng. test driven development)

## **2.3 Restruktuiranje - refactoring programskog kôda**

Razvoj softvera se realizira pisanjem izvornog kôda. Svaka funkcija softvera se može na više načina izraziti izvornim kôdom. Refactoring je proces izmjene strukture izvornog kôda **bez promjene** funkcionalnosti. Svrha refactoringa je učiniti kôd preglednijim, smanjiti mogućnost grešaka, optimizirati korištenje računarskih resursa ili učiniti izvođenje bržim.

## **2.4 Programiranje u paru**

Programiranje u paru je tehnika u kojoj jedan developer sjedi za tastaturom (“vozač”, eng. driver), a drugi sjedi pored njega (“posmatrač”, eng. observer) i prati na ekraru njegov unos. U toku kucanja programskog kôda, svaki put kada se pojavi nejasnoća ili dilema, developeri započinju diskusiju. Primjenom ove tehnike postižu se sljedeći efekti:

- “posmatrač” na licu mjesta vrši pregled i kontrolu kvaliteta i čitljivosti programskog kôda (eng. code review)
- izvorni kôd je u startu zajednička, a ne individualna vrijednost
- stalno se dešava intenzivna i kvalitetna komunikacija unutar razvojnog tima
- članovi razvojnog tima dijele i prenose znanje unutar tima (eng. sharing knowledge)

## **2.5 Testiranje na prvom mjestu (TDD)**

Razvoj vođen testiranjem (eng. Test Driven Development) je praksa u pisanju softvera kojom se pisanje započinje pisanjem testnih funkcija. Na taj način izvorni kôd je u potpunosti pokriven setom testova. Testovi imaju sljedeće funkcije:

- Testovi dokumentuju namjere izvornog kôda na najbolji način
- Izrada testnih funkcija je proces koji na najboljli način kombinuje razvoj i dizajn. Prije implementacije developer može vidjeti kako će kôd biti korišten
- Refactoring testabilnog kôda je efikasan i jednostavan, sa minimalnim rizicima

### **2.5.1 Rust TDD i refactoring primjer**

U ovom primjeru demonstrirane su prakse TDD-a i refactoringa na primjeru rješavanja problema projekta Euler<sup>2</sup> sa rust programskim jezikom<sup>3</sup>.

#### **[Priča 1]**

##### **Euler problem 1:**

- Naći sumu svih prirodnih brojeva koji su djeljivi bez ostatka sa 3 i 5, manji od zadatog broja N.
- Ako se za graničnu vrijednost postavi 10, rezultat je 23.

*Listing 1: Korisnička priča "Euler problem 1"*

<sup>2</sup> <https://projecteuler.net/problem=1>

<sup>3</sup> <https://hackernoon.com/a-taste-of-rust-6d8fc60e050> ; [https://github.com/hernad/rust\\_tdd](https://github.com/hernad/rust_tdd)

Opis problema ujedno predstavlja prvu korisničku priču programa (Listing 1). Iz nje se direktno može definisati prva testna funkcija (Listing 2):

```
#[test]
fn euler_problem_1_test_1() {
    assert_eq!(euler_problem_1(10), 23 );
}
```

*Listing 2*

Da bi se program mogao kompajlirati, treba dodati praznu funkciju - skeleton (Listing 3) :

```
fn euler_problem_1(max: u64) -> u64 {
    unimplemented!();
}
```

*Listing 3*

Prvo pokretanje testa daje očekivano negativan rezultat (Listing 4):

```
$ cargo test
  Finished dev [unoptimized + debuginfo] target(s) in 0.01s
    Running target\debug\deps\rust_tdd-0825df093aadf3ef.exe

running 1 test
test euler_problem_1_test_1 ... FAILED

failures:
---- euler_problem_1_test_1 stdout ----
thread 'euler_problem_1_test_1' panicked at 'not yet implemented', src\main.rs:7:5
note: Run with `RUST_BACKTRACE=1` for a backtrace.

failures:
    euler_problem_1_test_1

test result: FAILED. 0 passed; 1 failed; 0 ignored; 0 measured; 0 filtered out
error: test failed, to rerun pass '--bin rust_tdd'
```

*Listing 4*

Negativan rezultat testa se prikazuje crvenom bojom, uspješan zelenom. Tok razvoja tokom koga programer traži rješenje u kome će svi testovi postati “zeleni” je standardna rutina TDD procesa. Jednostavnost, vizuelna reprezentacija (crveno/zeleno) su veoma značajne za efikasnost i kvalitet procesa. Developer postiže maksimalni **fokus** i produktivnost u razvojnog ciklusu. Listing 5. prikazuje prvu implementaciju:

```
fn euler_problem_1(_max: u64) -> u64 {
    let mut result = 1;
    let mut i = 0;
    loop {
        if i >= _max {
            break;
        }
        if i % 3 == 0 || i % 5 == 0 {
            result += i;
        }
        i += 1;
    }
    result
}
```

*Listing 5*

Slijedi test ispravnosti (Listing 6):

```
$ cargo test

running 1 test
test euler_problem_1_test_1 ... FAILED

failures:

---- euler_problem_1_test_1 stdout ----
thread 'euler_problem_1_test_1' panicked at 'assertion failed: `(left == right)`
  left: `24`,
  right: `23`', src\main.rs:3:5
note: Run with `RUST_BACKTRACE=1` for a backtrace.

failures:
  euler_problem_1_test_1

test result: FAILED. 0 passed; 1 failed; 0 ignored; 0 measured; 0 filtered out
error: test failed, to rerun pass '--bin rust_tdd'
```

*Listing 6*

Rezultat testa je negativan. Izvještaj kaže da da je lijeva strana testa dala rezultat 24, umjesto željenog 23. Ove informacije pomažu da se utvrdi uzrok greške, u ovom slučaju to je pogrešna inicijalna vrijednost varijable "result". Nakon ispravke linije "`let mut result = 0;`", te ponavljanje testne procedure, po prvi put se dolazi do ispravne implementacije tražene funkcije (Listing 7).

```
running 1 test
test euler_problem_1_test_1 ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

*Listing 7*

Ovakav način razvoja (eng. development workflow) sadrži bitne motivacijske elemente: uspješnost testa stvara osjećaj postignuća kod developera, te daje dodatno motivaciju da se nastavi na podizanju kvalitete programskog kôda.

Željeni rezultat je postignut, ali je funkcija komplikovana. Jednostavnije rješenje je prikazano na Listingu 8:

```
fn euler_problem_1( _max: u64 ) -> u64 {

    let mut result = 0;
    for i in 0.._max {
        if i % 3 == 0 || i % 5 == 0 {
            result += i;
        }
    }
    result;
}
```

*Listing 8*

Testiranje kaže da je nova iteracija ispravna, ujedno i prvi uspješan refactoring! Međutim, postojeći kôd je napisan proceduralno. Rust nudi napredne funkcionalne koncepte koji su iskorišteni u narednoj iteraciji (Listing 9):

```
fn euler_problem_1( _max: u64 ) -> u64 {  
    let mut result = 0;  
    for i in (0.._max).filter(|n| n % 3 == 0 || n % 5 == 0) {  
        result += i;  
    }  
    result  
}
```

Listing 9

Pošto je `(0 .. _max)` rang koji implementira “Iterator” interfejs (jezikom “rust”-a “trait”; bos. osobina), na njega se mogu primjeniti iteratorske funkcije koje za argumente uzimaju anonimne funkcije (eng. closure). Zato je iskorištena funkcija “filter”, čiji je argument anonimna funkcija `|n| n % 3 == 0 || n % 5 == 0` koja ispituje djeljivost argumenta. Ponovljeni test daje pozitivan rezultat, što govori da je i ovaj refactoring uspješan. Variable koje mijenjaju vrijednost<sup>4</sup> nisu poželjne u funkcionalnom modelu programiranja. Funkcionalno programiranje bazira se na paradigmi formiranja lanca funkcionalnih transformacija, po uzoru na matematičke transformacije “čistih” funkcija (funkcije koje nemaju sporedna dejstva; eng. “side effects”):

```
f1( arg ) -> f2(resultat_f1) -> ... > fn( resultat_fn-1 ) -> resultat_n
```

Funkcionalni koncept je do kraja realiziran dodavanjem `sum()` funkcije (Listing 10):

```
fn euler_problem_1( _max: u64 ) -> u64 {  
    (0.._max).filter(|n| n % 3 == 0 || n % 5 == 0).sum()  
}
```

Listing 10

U skladu sa uobičajenom rutinom pokreće se test ispravnosti nove implementacije (Listing 11):

```
running 1 test  
test euler_problem_1_test_1 ... ok  
  
test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

Listing 11

U nekoliko iteracija postignuto je kvalitetno rješenje. U tom procesu glavni “pomoćnik” bila je serija testova.

---

<sup>4</sup> “rust” zahtjeva da se mutirajuće variable eksplicitno označavaju sa rezervisanim riječi “mut”

U nastavku je prikazana još jedna karakteristična situacija unutar XP tima. Razvojnom timu je pridružen član koji zastupa klijenta. On je poslovni analitičar. Njegov glavni alat je aplikacija za tabelarne proračune.

Analitičar dolazi sa proračunom Eulerove funkcije sa graničnim elementom 21. Traži od razvojnog tima da provjeri da li postojeća implementacija ispravno proračunava i taj slučaj (Slika 1):

A	B	C	D	E	F	G	H	I	J	K	L
1	1	0									
2	2	0									
3	3	2									
4	4	0									
5	5	4									
6	6	4									
7	7	0									
8	8	0									
9	9	6									
10	10	10									
11	11	0									
12	12	12									
13	13	0									
14	14	0									
15	15	14									
16	16	0									
17	17	0									
18	18	18									
19	19	0									
20	20	20									
21	98										
22											
23											

Slika 1: Poračun "Eulerove funkcije" u "libreoffice calc"

Uvodi se nova testna funkcija (Listing 12):

```
#[test]
fn euler_problem_1_test_2() {
    assert_eq!(euler_problem_1(21), 98 );
}
```

Listing 12

Svaka promjena izvornog kôda se kontroliše procedurom testiranja (Listing 13):

```
$ cargo test
Compiling rust_tdd v0.1.0 (file:///D:/devel/rust_tdd)
  Finished dev [unoptimized + debuginfo] target(s) in 0.83s
    Running target\debug\deps\rust_tdd-0825df093aadf3ef.exe

running 2 tests
test euler_problem_1_test_1 ... ok
test euler_problem_1_test_2 ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

Listing 13

Oba testa su "ok" ("zeleni"), čime je potvrđeno da je zadatak uspješno završen.

### **3 “Scrum”**

#### **3.1 Uvod**

“Scrum” je termin u ragbiju koji se koristi za formaciju igrača kojom se započinje igra nakon prekida (Slika 2), u našoj terminologiji poznat kao “bula”.



Slika 2: Ragbi igrači u "buli" [wikipedia]

Osnovna paradigma koja definiše ovu metodu razvoja je prihvatanje principa neizvjesnosti, odnosno nemogućnosti razumjevanja svakog detalja problema i potpuno tačnog predviđanja posljedica u toku procesa. Stoga se “scrum” fokusira na maksimiziranje timskog potencijala da efektivno i efikasno izvršava zadatke, fleksibilno pristupa zahtjevima koji se pojavljuju tokom procesa, te se prilagođava razvoju tehnologije i zahtjevima tržišta.

#### **3.2 Dnevni “Stand-up” sastanci (eng. stand-up meetings)**

Dnevni “Stand-up” sastanci su efektivan način da se članovi tima međusobno informišu o stanju projekta. Kao što ime sugerire, učesnici svoje izlaganje izlažu stojeći. Taj način izlaganja pomaže da sastanci budu kratki. Svakom učesniku se daje kratak period vremena za izlaganje.[APRAT]

Da bi sastanak imao potreban fokus, svaki učesnik odgovara na tri pitanja:

- Šta sam postigao juče?
- Šta planiram za danas?
- Koji su moje glavne prepreke i problemi da realiziram planirano?

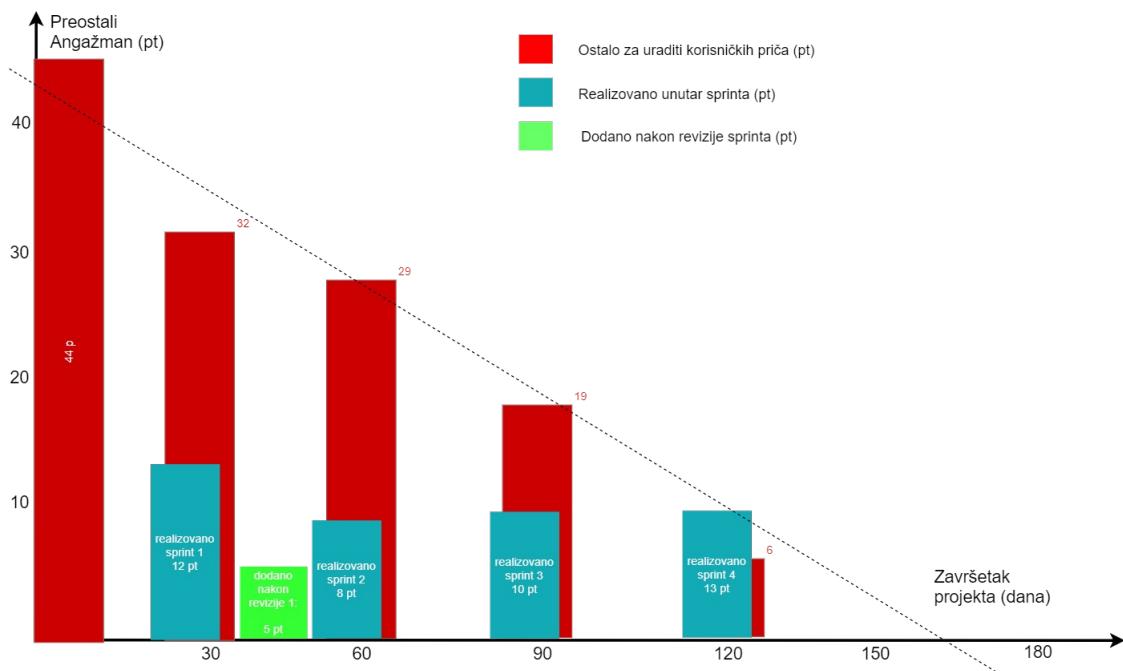
### **3.3 “Sprintovi”**

Ekstremno programiranje uvodi koncept iteracije kao jedinicu razvoja softvera. “Scrum” svoje iteracije naziva “sprintovi”. Pojedinačni sprint traje 15-30 dana. Vremensko ograničenje sprinta (eng. timeboxing) je obavezna praksa. Ako desi da planirane funkcije nije moguće završiti unutar sprinta, određuju se one koje se izbacuju za naredni sprint da bi se ispoštovalo vremensko ograničenje. Kontinuirana vremenska dinamika, striktnost u trajanju sprintova vođi tima (eng. Scrum master) omogućava da nakon nekoliko sprintova pouzdano utvrdi **kapacitet** tima (eng. team velocity). Ta metrika je ključna informacija za planiranje narednih sprintova, i naravno završetak cijelog projekta.

#### **3.3.1 Centralna lista korisničkih priča (eng. Product backlog)**

Centralna lista korisničkih priča (eng. Product backlog) je kolekcija priča (funkcija) koje korisnik želi imati u svom softveru.

Ona je **prioritizirana** od strane **korisnika**, te **procijenjena** (u smislu napora - potrebnog angažmana, eng. effort) od strane **razvojnog tima**[ASAM]. Dinamika realizacije projekta prati se uz pomoć “Burn-down” dijagrama (Dijagram 3):



Dijagram 3: "Burn-down" dijagram

#### **3.3.2 Lista priča novog sprinta (eng. Spring backlog)**

Iz centralne liste se na osnovu prioriteta i procjena angažmana formira lista priča (eng. Sprint backlog) koje će se realizovati unutar narednog sprinta. Lista za naredni sprint se utvrđuje nakon revizije neposredno završenog sprinta. Nakon revizije “Scrum master” ima svježe informacije o kapacitetu tima, te povratne informacije od korisnika. Te informacije mu pomažu da formira listu zadataka za realizaciju koju je tim **sposoban** realizovati, a korisnik najviše treba i očekuje.

## 4 “Lean software development”

### 4.1 Uvod

Ova metoda razvoja softvera se temelji na “Lean” metodama u oblasti proizvodnje, logistike i administrativnih operacija unutar finansijskih i trgovinskih organizacija<sup>5</sup>.

Doslovni prevod riječi “lean” sa engleskog znači mršav, vitak - bez viška kilograma. Termin asocira na glavni cilj ove metode: isporučiti korisniku vrijednost što brže eliminisanjem beskorisnog otpada i podizanjem kvalitete usluga i/ili proizvoda. “Lean” metodologija je bazirana na sedam praksi (Dijagram 4):



Dijagram 4: "Lean software development" prakse

### 4.2 Eliminacija otpada

Otpad je sve što ne doprinosi vrijednosti finalnog proizvoda:

- Nepotrebna ili zastarjela dokumentacija
- Funkcije softvera koje niko ne koristi
- Neefikasne ili nepotrebne operacije i procesi

### 4.3 Kvalitet izrade

Kvalitetnom izradom softverskog proizvoda preveniraju se sofverske greške, te eliminišu operacije potrebne da se greške otklone nakon stavljanja u upotrebu.

5 <https://dzone.com/refcardz/getting-started-lean-software>

### **4.3.1 Kreiranje znanja**

Saznanja stečena u prethodnim operacijama su baza znanja za naredne razvojne cikluse. To znanje postaje “gorivo” - akcelerator razvojnog tima.

### **4.3.2 Donošenje odluka u najboljem trenutku**

Znanje se stiče u svakoj fazi rada. Što vrijeme odmiče, tim posjeduje više informacija i znanja za pravilno odlučivanje. Zato strateške odluke (npr. arhitektura rješenja u narednoj iteraciji) treba donijeti u najboljem trenutku - što je moguće kasnije.

### **4.3.3 Brze isporuke**

Kada se dođe do određenog rješenja, ne treba odlagati sa njegovom isporukom. Rješenje koje je aktuelno u jednom trenutku, nakon određenog perioda može izgubiti na vrijednosti. Dodatno, u periodu neisporuke (bez opravdanog razloga) razvojni tim ne može dobiti povratne informacije od korisnika. Nedostatak informacija onemogućava rast znanja o proizvodu na kome se radi.

### **4.3.4 Uvažavanje ljudi**

Ljudi su ti koji obezbeđuju razvoj (developeri) i ocjenjuju vrijednost softvera koji se za njih pravi (korisnici). Zato su komunikacija bazirana na uvažavanju i stvaranje dobrih uslova rada za razvojni tim obavezni elementi projekta organiziranog po “Lean” principima.

### **4.3.5 Optimiziranje cjeline**

Posmatranje projekta u cijelini (eng. “see the whole”, “big picture”) je princip djelovanja i razmišljanja o projektu koji garantuje da sve aktivnosti koje se poduzimaju vode ka najboljem mogućem rezultatu. Ova praksa pomaže da sve do sada navedene prakse rezultiraju sinergijskim učinkom.

# 5 “Kanban”

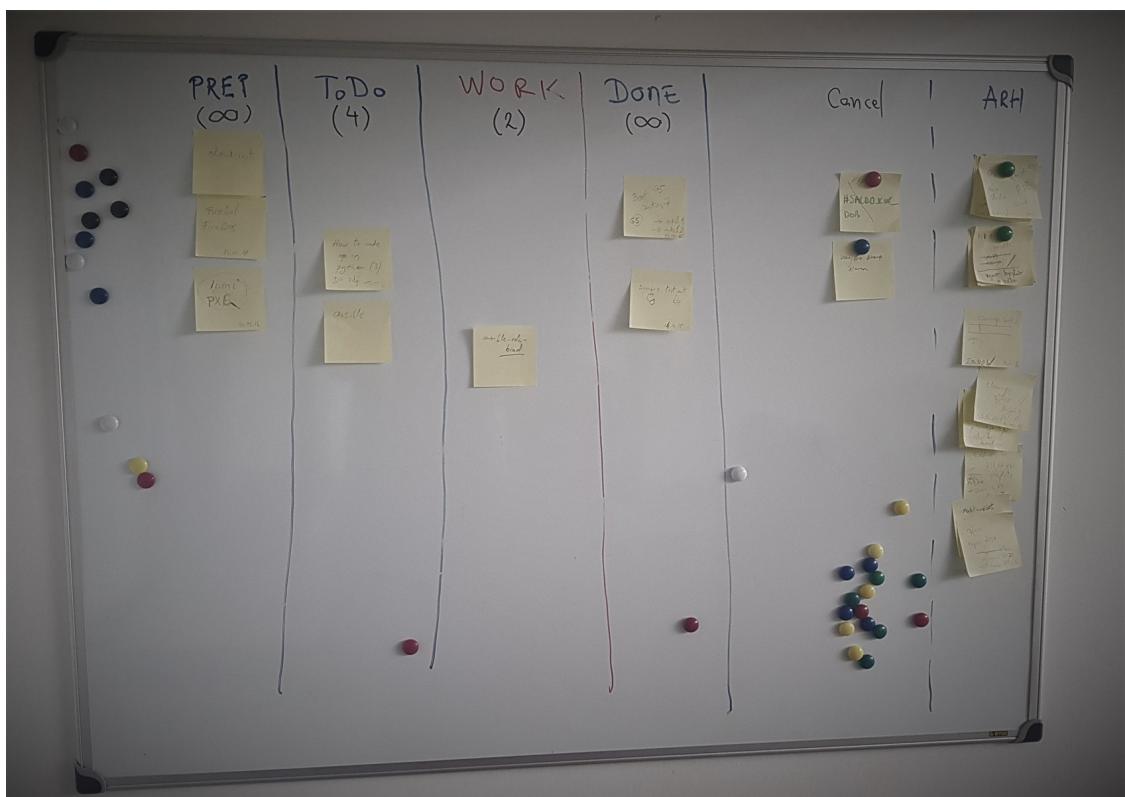
## 5.1 Uvod

“Kanban” u prevodu sa japanskog znači “pano”, “oglasna tabla”. “Kanban” je sistem za planiranje proizvodnje kao podrška za “lean manufacturing” i “just-in-time manufacturing”. Inžinjer u “Toyoti” Taiichi Ohno je razvio kanban metodu da bi povećao efikasnost proizvodnje<sup>6</sup>.

U oblasti razvoja softvera “Kanban” pripada grupi “Lean” metoda, u kojoj je “Kanban” ploča centralno mjesto vizuelnog prikaza stanja i planiranja toka operacija. “Kanban” je veoma podesan u poslovima podrške korisnicima.

“Kanban” omogućava timu da bolje razumije operacije u toku. Limitiranje broja operacija obezbjeđuje redukciju otpada nastalog uporednim obavljanjem više operacija uslijed promjene konteksta (eng. multitasking and context switching waste)<sup>7</sup>. Limitiranjem se takođe obezbjeđuje fokus na operacije koje će otkočiti dovođenje operacija do krajnjeg cilja.

## 5.2 “Kanban” ploča



Slika 3: Kanban ploča u preduzeću "bring.out"

6 <https://en.wikipedia.org/wiki/Kanban>

7 [https://en.wikipedia.org/wiki/Kanban\\_\(development\)](https://en.wikipedia.org/wiki/Kanban_(development))

Na slici 3. je prikazana “kanban” ploča koja se koristi za poslove podrške korisnicima. Ono što je vrlo važno uočiti jeste da svaki tim organizuje “kanban” ploču u skladu sa svojim načinom rješavanja zadataka. Gornja ploča sadrži sljedeće kolone (Tabela 1):

Oznaka kolone	Opis	Limit	Objašnjenje
PREP	Priprema (eng. preparation)	$\infty$	Pripremne radnje potrebne da bi se moglo pristupiti rješavanju
ToDo	U redu čekanja (eng. to do)	4	Zadaci iz pripreme se stavljaju u skladu sa prioritetima u “ToDo” listu. Ova lista je ograničena na 4 stavke. Razlog preglednost, kao i činjenica da je naredna, radna kolona limitirana na dvije stavke
Work	U realizaciji	2	Zadaci u realizaciji. Limit od dvije stavke određen je trenutnim kapacitetom tima.
Done	Realizovano	$\infty$	Zadaci koji su realizovani
Cancel	Otkazano	-	Otkazani zadaci se drže u ovoj koloni ograničeni period, dok postoji mogućnost da se zadatak vrati (u istoj ili promjenjenoj formi) u rad
Arh	Arhiva	-	Arhiva realizovanih aktivnosti. Na kraju određenog perioda, u našem slučaju na godišnjem nivou se “čisti” - prazni.

*Tabela 1: Značenje kolona na “kanban” ploči*

Organizacija “kanban” ploče se može mijenjati u skladu sa potrebama i stanjem tima. Npr. u slučaju da se kapacitet tima poveća, limit na “ToDo” i “Work” kolonama se može povećati. Ukoliko se pokaže da ploča nije dovoljno informativna, može se dodati nova kolona ili izbrisati postojeća. Kartice na ploči očekivano predstavljaju zadatake. Na karticama se navode osnovne informacije koje tim koristi u realizaciji:

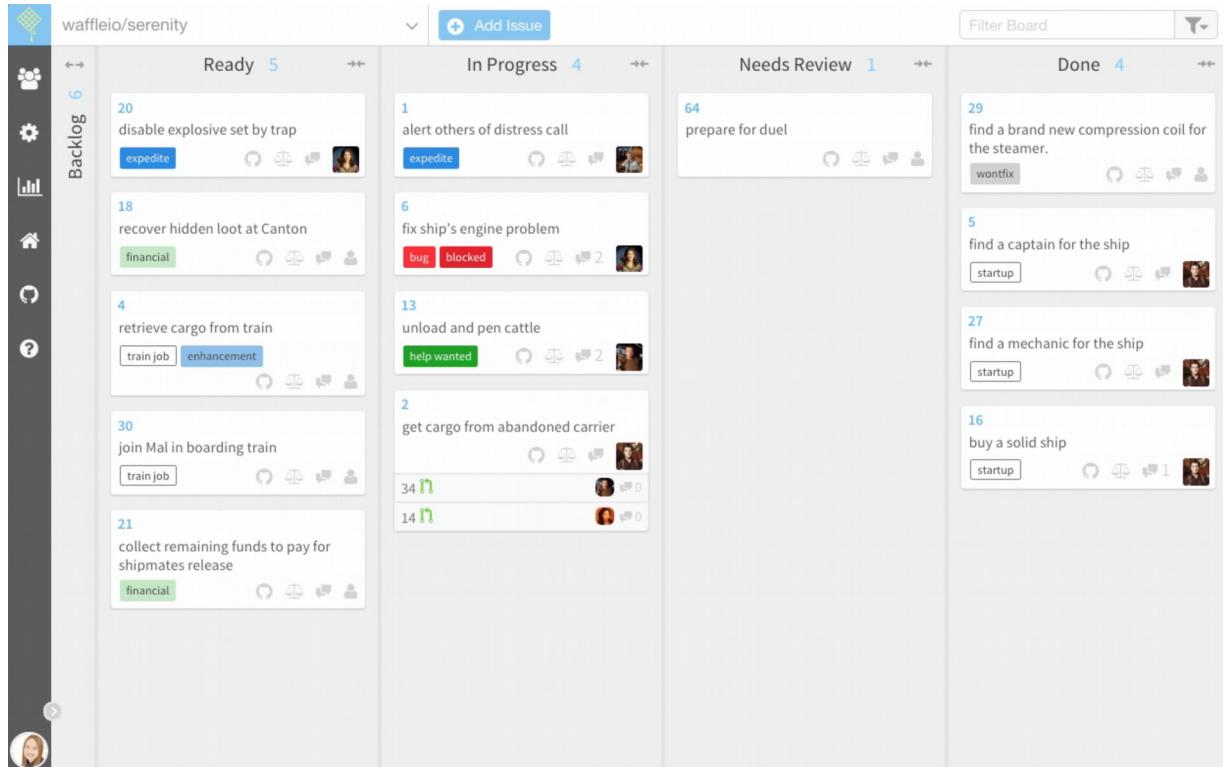
- Opis zadataka
- Datum zaprimanja zadataka
- Procjena potrebnog angažmana (u satima ili tačkama)
- Referentni broj u aplikaciji za praćenje korisničkih zahtjeva (eng. bug. tracker)
- Datum realizacije

Kartica se postupno ažurira potrebnim informacijama. Kartice se mogu označavati različitim bojama u skladu sa njihovim tretmanom u procesu rješavanja problema, npr:

- Crvena - kritične greške koje onemogućavaju rad korisnika
- Zelena - novi zahtjevi
- Narandžasta - prilagodbe korisničkog interfejsa
- Plava - rad na izvještajnim opcijama aplikacije

Ono što je najbitnije uočiti jeste da su organizacija kanban ploče i oznake na karticama kontekstualizirane načinom na koji tim koji rješava svoje zadatke. Ono što je za jedan tim bitno, za drugi može biti prenatrpano informacijama. Informativnost i preglednost koje obezbjeđuje vizuelna reprezentacija, te jednostavno ažuriranje informacija su ključni atributi ove metode razvoja.

“Kanban” ploča može biti virtualna ili fizička. Fizička ploča je podesna zato što je odličan medij za diskusije tima o poslovima koje treba obaviti. Kao takva, ona je idealno mjesto za organizovanje “stand-up” sastanaka. U slučaju da su članovi tima dislocirani, ili je kanban aplikacija već integrirana sa ostalim alatima koje tim svakodnevno koristi, umjesto fizičkog koristi se virtualna “kanban” ploča. Primjer takve aplikacije je waffle.io<sup>8</sup> (Slika 4):



Slika 4: Prikaz ekrana “waffle.io” aplikacije [github.com]

8 <https://github.com/waffleio/waffle.io>

## 6 Zajedničke prakse i koncepti

Svaka od navedenih metoda ima fokus na određene prakse i koncepte. Međutim, određene prakse su postale sastavni dio svake savremene metode razvoja softvera.

### 6.1 Upravljanje revizijama izvornog kôda (VCS)

Sa softverom za upravljanje verzijama izvornog kôda (eng. version control system - VCS) članovi razvojnog tima imaju uvid u istoriju promjena na softveru. Izvorni kôd, kao glavni artifakt softvera, ima posebno mjesto u razvojnom ciklusu. Gotovo sve aktivnosti u razvoju softvera u konačnici se završavaju pisanjem novog ili ispravkom postojećeg kôda.

*“Softver za kontrolu verzija izvornog kôda je sistem koji bilježi promjene na datoteci ili setu datoteka tokom vremena tako da se specifične verzije mogu naknadno vratiti.” [PROGIT]*

Baza promjena izvornog kôda je osnovni alat svakog developera. Prvi ovakvi alati su bili organizovani na klijent-server principu (CVS<sup>9</sup>, subversion<sup>10</sup>). Kod tih alata na lokalnoj strani (računaru developera) postoji samo jedna revizija - aktuelna revizija na kojoj radi. Za pristup ostalim revizijama mora se kontaktirati VCS server. Danas, međutim, dominiraju distribuirani VCS alati (Git<sup>11</sup>, Mercurial<sup>12</sup>). Iako “Mercurial” koristi nekoliko velikih kompanija i projekata (Facebook, Mozilla, openjdk), “Git” je danas najviše zastupljen<sup>13</sup>. Popularizaciji “Git”-a su značajno pomogli web servisi “Github”<sup>14</sup> i “Gitlab”<sup>15</sup>. Ovi servisi su mnogo više od VCS alata. Oni sadrže komponente koje obuhvataju kompletan razvojni ciklus softvera:

- Softver za kontrolu verzija (VCS)
- Generacija niza statistika vezanih za izvorni kôd
- Organizovanje razvojnog tima
- Praćenje grešaka i zahtjeva (eng. bug tracking)
- Automatska integracija binarnih verzija (eng. continuous integration)

Kod distribuiranog VCS softvera svaki developer ima sopstvenu kopiju distribuirane baze. On može da radi na svojoj kopiji bez pristupa centralnom serveru. Zato distribuirani način rada zahtjeva rješavanje kolizija prilikom spajanja verzija više developera koji su radili na istom kôdu (eng. merging and resolving conflict). Kod prvih verzija distribuiranih VCS alata ti algoritmi nisu bili usavršeni, pa su dominirala klijent-server rješenja. Međutim, Linus Torvalds je 2005, kao vođa Linux projekta izdao prvu verziju “git”-a. Linus je do tada za razvoj “Linux”-a koristio jedan komercijalni distribuirani VCS alat. Kada je dalje korištenje tog alata bilo nemoguće radi licencnih restrikcija, Linus nije prihvatio prijedlog da se razvoj Linuxa odabere neki centralizirani “opensource” VCS alat (što je u to vrijeme bila jedina opcija ako se ograniči na otvoreni softver), nego je odlučio napraviti sopstveni distribuirani VCS alat. U narednim godinama oko “Git”-a je razvijen čitav eko-sistem servisa i aplikacija.

Zanimljivost 1: Istorija git-a

9 [https://en.wikipedia.org/wiki/Concurrent\\_Versions\\_System](https://en.wikipedia.org/wiki/Concurrent_Versions_System)

10 [https://en.wikipedia.org/wiki/Apache\\_Subversion](https://en.wikipedia.org/wiki/Apache_Subversion)

11 <https://git-scm.com>

12 <https://www.mercurial-scm.org>

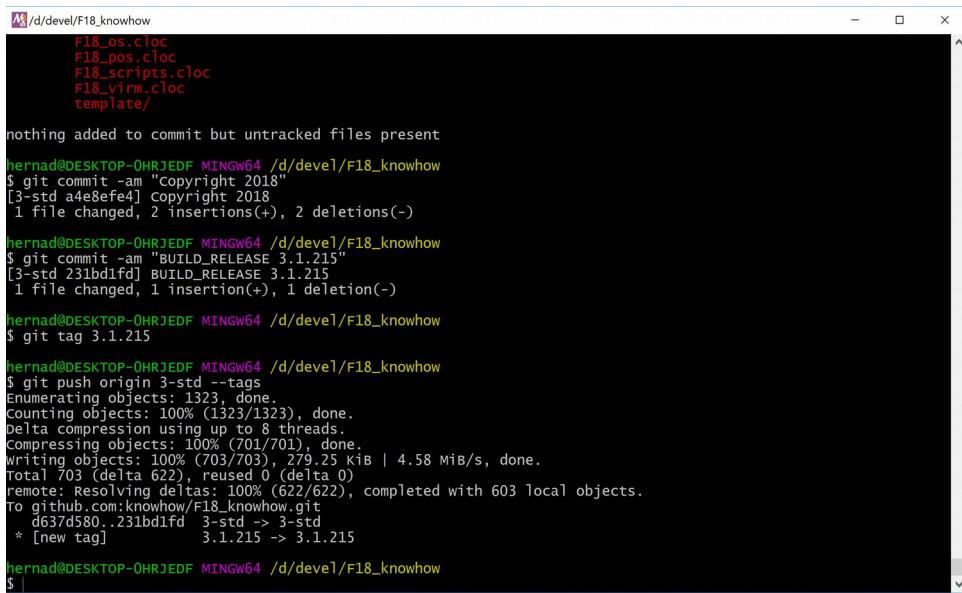
13 Zanimljivost: Istorija git-a

14 <https://github.com>

15 <https://gitlab.com>

- Dokumentacija softvera (wiki stranice)
- Komunikacija unutar razvojnog tima i sa korisnicima integracijom “chat” servisa (razgovor u realnom vremenu)
- Pregled izvornog kôda (eng. code review workflow)

Slike 5. i 6. prikazuju standardni način korištenja “git”-a.



```

/d/devel/F18_knowhow
F18_os.cloc
F18_pos.cloc
F18_scripts.cloc
F18_virm.cloc
template/

nothing added to commit but untracked files present

hernad@DESKTOP-0HRJEDF MINGW64 /d/devel/F18_knowhow
$ git commit -am "copyright 2018"
[3-std a4e8efed] copyright 2018
 1 file changed, 2 insertions(+), 2 deletions(-)

hernad@DESKTOP-0HRJEDF MINGW64 /d/devel/F18_knowhow
$ git commit -am "BUILD_RELEASE 3.1.215"
[3-std 231bd1fd] BUILD_RELEASE 3.1.215
 1 file changed, 1 insertion(+), 1 deletion(-)

hernad@DESKTOP-0HRJEDF MINGW64 /d/devel/F18_knowhow
$ git tag 3.1.215

hernad@DESKTOP-0HRJEDF MINGW64 /d/devel/F18_knowhow
$ git push origin 3-std:tags
Enumerating objects: 1323, done.
Counting objects: 100% (1323/1323), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (701/701), done.
Writing objects: 100% (703/703), 279.25 KiB | 4.58 MiB/s, done.
Total 703 (delta 622), reused 0 (delta 0)
remote: Resolving deltas: 100% (622/622), completed with 603 local objects.
To github.com:knowhow/F18_knowhow.git
  d637d580..231bd1fd 3-std > 3-std
 * [new tag]            3.1.215 -> 3.1.215

hernad@DESKTOP-0HRJEDF MINGW64 /d/devel/F18_knowhow
$ |

```

*Slika 5: Osnovne “git” operacije u konzolnom režimu - commit, tag, push*

```

$ git log -n 10
commit 91af01aef8622d9f785a7197fc5e6e2675c7cdd9 (HEAD -> 3-std)
Author: Ernad Husremovic <hernad@bring.out.ba>
Date:   Mon Aug 20 15:40:58 2018 +0200

F18 nalog count out

commit a689788b1577e463adb08ff854cfcc25c78bceec (origin/HEAD, origin/3-std)
Author: Ernad Husremovic <hernad@bring.out.ba>
Date:   Mon Aug 20 15:38:51 2018 +0200

unzbrka

commit 33b9be3ea73f5f72587d077a5e756283d5bbb154
Author: Ernad Husremovic <hernad@bring.out.ba>
Date:   Mon Aug 13 18:27:05 2018 +0200

.....
.....

```

*Slika 6: git log promjena, pregled u konzolnom režimu*

## 6.1.1 “Github” repozitorij

Na slikama 7, 8. i 9. dati su karakteristični prikazi “web” interfejsa jednog projekta<sup>16</sup> koji se hostira na servisu “github”:

The screenshot shows the GitHub repository page for 'F18\_knowhow'. At the top, there are navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, there's a search bar and a summary section with metrics: 8,376 commits, 10 branches, 1,040 releases, and 3 contributors. A list of recent commits is displayed, each with a small icon, the author's name ('hernad unzbrka'), the commit message, and the date. The commits are dated from 3 years ago to 10 days ago.

Slika 7: Github - Pregled datoteka

The screenshot shows the GitHub repository page for 'F18\_knowhow' with the 'Overview' tab selected. It displays a list of branches under 'Your branches' and 'Stale branches'. Each branch entry includes the branch name, the last update time, the author, and a 'New pull request' button. The 'Your branches' section shows branches like '3-std', '3-vind', '23100-1d', 'hcurl', and '23100-fakt'. The 'Stale branches' section shows branches like 'semaphores', '1.4', 'ng2', '1.7', and '23100-fakt'. There are also 'View more of your branches' and 'View more stale branches' links at the bottom.

Slika 8: Github - Pregled "branch"-ova projekta

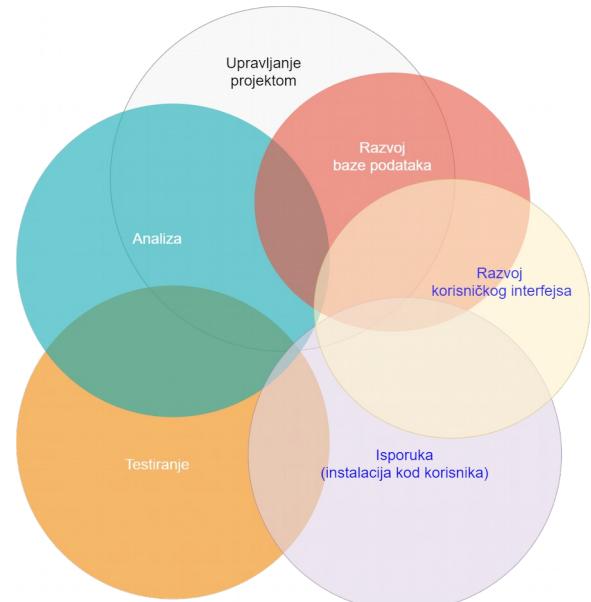
16 [https://github.com/knowhow/F18\\_knowhow](https://github.com/knowhow/F18_knowhow)

*Slika 9: Github: Pregled revizija (eng. commits)*

## 6.2 Multifunkcionalni tim

Agilne projekte karakteriše činjenica da uloge pojedinih članova nisu striktne. Članovi tima uzimaju svaki zadatak koji će doprinijeti projektu - bez obzira na titulu i glavnu ulogu u projektu[ASAM].

Naravno, odabir zadataka je primarno baziran na ekspertizi, glavnoj ulozi, preferencijama i efikasnosti članova. Međutim, ako određeni zadatak traži angažman cijelog tima da bi se postigao projektni cilj, svi se angažuju na rješavanju tog zadataka (Dijagram 5).



*Dijagram 5: Uloge članova tima postoje, ali nisu striktne*

### 6.3 Agilni trener

Agilni trener pomaže timu da se usvoje agilne prakse. On ili ona pomaže timu da usvoji agilni pristup rješavanju zadataka, te da pomogne otklanjanju mentalnih, emocionalnih i tehničkih barijera koje onemogućavaju tim da djeluje agilno[LEARNAG].



Slika 10: Trener američkog fudbala (dječja liga) daje upute članovima tima [pixabay.com]

Agilni trener je učitelj, kritičar, podrška i glavni motivator tima (Slika 10). On radi sa svakim članom tima na način da mu objašnjava ne samo kako da primjeni određene agilne prakse, nego i da razumije **zašto** to radi. Postoji niz primjera timova koji su nisu postigli nikakav značajan napredak primjenom agilnih metoda upravo zato što nisu promijenili način razmišljanja (eng. mindset). Multifunkcionalni tim, iterativni pristup, direktna komunikacija lice-u-lice, promjene u hijerarhijskoj strukturi tima, daće pozitivne efekte samo pod uslovom da članovi tima razumiju zašto funkcionišu na taj način. Razumjeti "zašto" je posebno bitno u organizacijama koje imaju dugogodišnje "tradicionalne" strukture i organizacije.

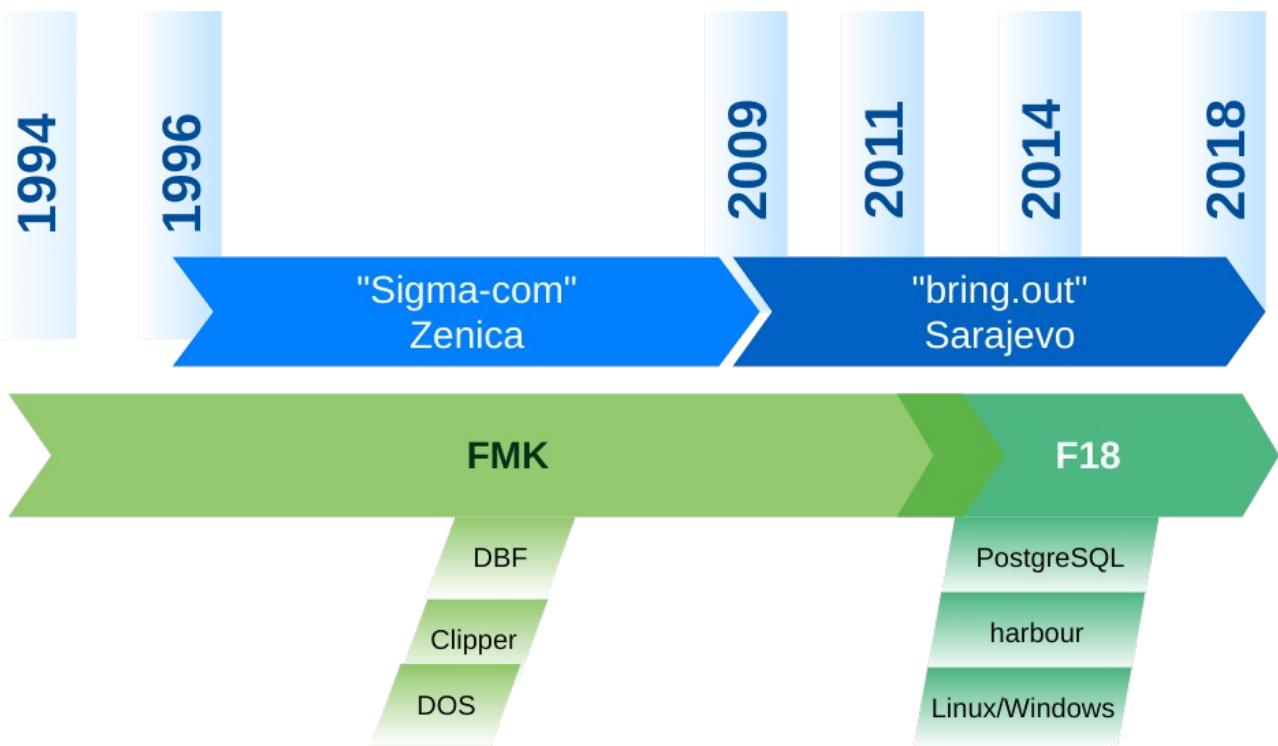
Svaka organizacija je određena kontekstom unutar koga djeluje. Taj kontekst kreiraju interne vrijednosti i odnosi unutar tima, korisnici, te pravila industrije za koju se problem rješava.

Agilni trener je sposoban da u konkretnom kontekstu primjeni odgovarajuću agilnu metodu. On dobro razumije agilne vrijednosti i principe na osnovu sopstvenih i iskustava organizacije drugih timova.

## 7 F18 - “knjigovodstvo za Bosance”

### 7.1 Uvod

F18 je knjigovodstvena aplikacija<sup>17</sup> preduzeća “bring.out” namjenjena za bosansko-hercegovačko tržište. Projekat je započet 2011 godine, a prve verzije objavljene 2012 godine. Aplikativni kôd međutim nije pisan ispočetka, nego je preuzet do “FMK”<sup>18</sup> projekta istog preduzeća. Prva verzija “FMK” je objavljena 1994 godine (Dijagram 6).



Dijagram 6: Istorija preduzeća i projekta

#### 7.1.1 Cilj projekta

Polazna osnova F18 bila je izrada aplikacije koja će u funkcionalnom smislu biti identična “FMK”, pri čemu će eliminisati tehnološka ograničenja svog prethodnika. “FMK” je imao sljedeće nedostatke:

- Aplikacija pisana u napuštenom programskom jeziku “Clipper”<sup>19</sup>, koji je pisan za DOS operativni sistem<sup>20</sup>.

17 [https://github.com/knowhow/F18\\_knowhow](https://github.com/knowhow/F18_knowhow)

18 <https://github.com/bringout-fmk>

19 [https://en.wikipedia.org/wiki/Clipper\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Clipper_(programming_language))

20 <https://en.wikipedia.org/wiki/DOS>; “FMK” je radio pod “Windows” OS kao emulirana 16-bitna aplikacija

- Klijentska aplikacija je sa bazom podataka (DBF tabele<sup>21</sup>) komunicirala “File server” režimu, što je činilo komunikaciju u višekorisničkom mrežnom radu neopuzdanom i sporom
- Fizičko ograničenje veličine DBF tabele je bilo 1GB

Korisnički interfejs FMK bio je karakterni - terminal. Iako je to veliki nedostatak sa stanovišta modernog korisnika, taj interfejs je ostavljen i u F18 iz sljedećih razloga:

- “F18” je koncipiran kao nasljednik “FMK” koji će postojeći korisnici moći nastaviti bez dodatne obuke
- Promjena korisničkog interfejsa ne bi bilo moguća bez velikih promjena, što je tražilo dodatne programerske resurse i vrijeme razvoja. Ništa od toga nije bilo dostupno, tako da je postojeći korisnički interfejs od početka uzet kao karakteristika F18, a ne “bug”.
- “F18” je u vrijeme početnog razvoja tretiran kao privremeni projekat koji će biti zamjenjen “xTuple-Postbooks open-source” softverom<sup>22</sup> lokaliziranim za bosansko-hercegovačko tržište.

“F18” je klasična dvoslojna aplikacija za manipulaciju SQL bazom podataka u kojoj se većina logike nalazi na strani klijenta.

### **7.1.2 “Ad-hoc” rješenja, dugoročne glavobolje**

Stabilnost aplikacije u sloju pristupa bazi podataka je i nakon dvije godine razvoja bila upitna. Pokazalo se da je taj sloj aplikacije, napisan na brzinu, loše koncipiran. Funkcija ovog sloja, nazvanog “semafori”<sup>23</sup>, je bila obezbjediti da aplikativni kôd profunkcioniše nepromjenjen na način da se i dalje lokalno pristupa DBF tabelama koje se u pozadini sinhroniziraju sa SQL bazom. U preliminarnim testovima “semafori” su funkcionalisali dobro. Međutim, stavljanjem u produkciju pokazalo se da je ovaj koncept promašaj u slučajevima većih baza i višekorisničkog pristupa.

## **7.2 Preduzeće “bring.out”**

Preduzeće “bring.out” sa sjedištem u Sarajevu je osnovano u Zenici pod imenom “Sigma-com” 1996 godine. Od početka, bazni proizvod je bio set knjigovodstvenih aplikacija. “bring.out” danas nudi Linux serverska rješenja i knjigovodstvenu aplikaciju F18. Sva programska rješenja “bring.out” su “open-source” softver (nadalje OSS)<sup>24</sup>. “F18” je OSS softver objavljen pod CPAL<sup>25</sup> licencom.

### **7.2.1 Razvojni period, developeri**

U periodu 1994-2018 u razvoju su učestvovala četiri developera. Autor ovog rada je osnivač i prvi developer “FMK”, kao i “F18” projekta. Većinu vremena autor ipak nije bio uključen u programerske aktivnosti, nego je obavljao druge poslove.

---

21 <https://en.wikipedia.org/wiki/DBF>

22 <https://sourceforge.net/projects/postbooks> ; <https://github.com/knowhow/xtuple>

23 [https://github.com/knowhow/F18\\_knowhow/blob/1.7/common/semaphores.prg](https://github.com/knowhow/F18_knowhow/blob/1.7/common/semaphores.prg)

24 [https://en.wikipedia.org/wiki/Open-source\\_software](https://en.wikipedia.org/wiki/Open-source_software)

25 <https://opensource.org/licenses/CPAL-1.0>

## 7.2.2 Kriza projekta

Krajem 2014. godine projekat se našao pred velikim izazovom. Kolega koji je više od deset posljednjih godina bio glavni developer je prestao raditi u preduzeću. Nakon dugogodišnje pauze, autor je preuzeo poslove razvoja. Rezultati u tom periodu nisu bili ohrabrujući. Izvorni kôd je bio nečitljiv, neodržavan, dupliran, nedoslijedan, fragilan. Globalne i privatne varijable koje su izazivale “*Variable not found*” greške nakon svake promjene na kôd-u. Svaka promjena na projektu je bila frustracija za korisnike. Prijave grešaka: “*Ovo je radilo, sada više ne radi?!*” bile su svakodnevna pojava. Godine stihiskog i nesistematskog rada na projektu su uzele svoj danak. U preduzeću se počela voditi diskusija da li je rad na projektu uopšte održiv.

## 7.3 Korištene tehnologije

Iskorištenje postojećeg aplikativnog kôda, multiplatformski pristup te razvoj i korištenje OSS softvera uticale su na odabir tehnologija za razvoj F18.

### 7.3.1 Harbour programski jezik

Harbour<sup>26</sup> je OSS programski jezik koji je kompatibilan sa “Clipper” programskim jezikom u kome je napisan “FMK”. Harbour pripada familiji dinamičkih jezika kao što su “python” i “ruby”. Nije stekao popularnost van kruga bivših “Clipper” programera. Jezik podržava proceduralni i objektno-orientisani model programiranja. S obzirom da je “FMK” u većini pisan proceduralno, unutar F18 projekta se zadržao proceduralni model. “Harbour” pripada familiji xBase<sup>27</sup> jezika. Xbase jezici sadrže integriran jezik za manipulaciju tabelama. Primjer kreiranja dbf tabele (Listing 14):

```
LOCAL nI, aStruct := { ;  
    { "ID",          "N",   8, 0 }, ;  
    { "IME",          "C",  50, 0 }, ;  
    { "PREZIME",      "C",  50, 0 }, ;  
    { "GODINA_RADA", "N",   6, 0 }, ;  
    { "NETO_PLATA",  "N",   8, 2 }, ;  
    { "ZAPOSLEN",     "D",   8, 0 }, ;  
    { "OZENJEN",       "L",   1, 0 }, ;  
    { "MEMO1",         "M",  10, 0 }, ;  
    { "MEMO2",         "M",  10, 0 } }  
  
REQUEST DBFCDX  
dbCreate( "plate.dbf", aStruct, "DBFCDX", .T., "PLATE" )
```

Listing 14

Primjer otvaranja i sekvenčne pretraga (Listing 15):

```
? "LOCATE FOR PLATE->OZENJEN .AND. PLATE->NETO_PLATA > 1000"  
WAIT  
dbUseArea( , , "plate.dbf", "PLATE" )  
LOCATE for PLATE->OZENJEN .AND. PLATE->NETO_PLATA > 1000  
DO WHILE PLATE->( Found() )  
    ? PLATE->PREZIME, PLATE->IME, TESTDBF->OZENJEN, PLATE->NETO_PLATA  
    // primjer outputa: HUSREMOVIC ERNAD .T. 1100  
    CONTINUE  
ENDDO
```

Listing 15

26 [https://en.wikipedia.org/wiki/Harbour\\_\(software\)](https://en.wikipedia.org/wiki/Harbour_(software))

27 <https://en.wikipedia.org/wiki/XBase>

### 7.3.2 Pristup PostgreSQL bazi iz “harbour”-a

DBF tabele i xBase jezik za manipulaciju podacima nema previše sličnosti sa SQL-om što otežava rad sa relacijskim bazama podataka unutar “harbour”-a. Međutim, “harbour” podržava izradu različitih dodataka za pristup podacima (eng. replacable database drivers - RDD) koji obezbeđuju da se xBase model rada sa podacima primjeni na različite izvore podataka. Za potrebe pristupa PostgreSQL-u bazama se koristi PgSQL RDD<sup>28</sup>. Bazne funkcije su realizovane unutar F18 “core\_sql” komponenti<sup>29</sup>. Te funkcije omogućavaju da se bazi podataka pristupa idiomatski harbour/xBase programskom jeziku.

Primjer: U nizu izvještaja (kartica, lager lista faktura) za pristup fakturama iz određenog perioda korištena je sekvenca kôda prikazana na Listingu 16:

```
find_fakt_za_period( cIdFirma, dDatOd, dDatDo, NIL, NIL, "3" )

? "Fakture za period:", dDatOd, dDatDo
DO WHILE !EOF()
    cIdFirma := fakt->idfirma
    cBrDok := fakt->brdok
    cIdTipDok := fakt->idTipDok

    ? "Fakturna:" cIdFirma, cIdTipDok, cBrDok, " ima sljedeće stavke"
    DO WHILE !EOF .AND. fakt->idfirma == cIdFirma .AND. fakt->idtipdok == cIdTipok ;
        .AND. fakt->brdok == cBrDok
        ? "datum", fakt->datdok, "iznos:", fakt->cijena * fakt->kolicina
    ENDDO

ENDDO
```

Listing 16

Funkcija *find\_fakt\_za\_period()*<sup>30</sup> je realizovana uz pomoć core\_sql funkcije *use\_sql()*<sup>31</sup>:

```
FUNCTION find_fakt_za_period( cIdFirma, dDatOd, dDatDo, cOrderBy, cWhere, cTag )

LOCAL hParams := hb_Hash()
hb_default( @cOrderBy, "idFirma,idtipdok,brdok,rbr" )

IF cIdFirma <> NIL
    IF !Empty( cIdFirma )
        hParams[ "idfirma" ] := cIdFirma
    ENDIF
ENDIF
IF dDatOd != NIL
    hParams[ "dat_od" ] := dDatOd
ENDIF
IF dDatDo != NIL
    hParams[ "dat_do" ] := dDatDo
ENDIF
```

Listing 17 : *find\_fakt\_za\_period* implementacija (1/3)

28 <https://github.com/hernad/harbour-core/tree/my-master/contrib/hbpgsql> ;  
<https://github.com/hernad/harbour-core/tree/my-master/contrib/sddpg> ; PostgreSQL baza podataka je odabrana iz razloga što je ista baza korištena u “xTuple-Postbooks” projektu.

29 [https://github.com/knowhow/F18\\_knowhow/tree/3-std/core\\_sql](https://github.com/knowhow/F18_knowhow/tree/3-std/core_sql)

30 [https://github.com/knowhow/F18\\_knowhow/blob/3-std/fakt/\\_fakt\\_sql.prg](https://github.com/knowhow/F18_knowhow/blob/3-std/fakt/_fakt_sql.prg) ; Listinzi 17-19

31 [https://github.com/knowhow/F18\\_knowhow/blob/3-std/core\\_sql/use\\_sql.prg#L159](https://github.com/knowhow/F18_knowhow/blob/3-std/core_sql/use_sql.prg#L159)

```

hParams[ "order_by" ] := cOrderBy
hParams[ "indeks" ] := .T.

IF cTag == NIL
  cTag := "1"
ENDIF
hParams[ "tag" ] := cTag

IF cWhere != NIL
  hParams[ "where" ] := cWhere
ENDIF

IF !use_sql_fakt( hParams )
  RETURN .F.
ENDIF

GO TOP

RETURN ! Eof()

FUNCTION use_sql_fakt( hParams )
  LOCAL cTable := "fakt_fakt", cAlias := "FAKT"
  LOCAL cWhere, cOrder
  LOCAL cSql
  LOCAL hIndexes, cKey, cTag := "1"
  default_if_nil( @hParams, hb_Hash() )

  cSql := "SELECT * from fmk." + cTable
  cWhere := use_sql_fakt_where( hParams )
  cOrder := use_sql_fakt_order( hParams )

  IF !Empty( cWhere )
    cSql += " WHERE " + cWhere
    IF !Empty( cOrder )
      cSql += cOrder
    ENDIF
  ELSE
    cSql += " OFFSET 0 LIMIT 1"
  ENDIF

  IF hb_HHasKey( hParams, "alias" )
    cTable := hParams[ "alias" ]
  ENDIF
  IF hb_HHasKey( hParams, "tag" )
    cTag := hParams[ "tag" ]
  ENDIF

  SELECT ( F_FAKT )
  IF !use_sql( cTable, cSql, cAlias )
    RETURN .F.
  ENDIF

  IF hParams[ "indeks" ]
    hIndexes := h_fakt_fakt_indexes()
    FOR EACH cKey IN hIndexes:Keys
      INDEX ON &( hIndexes[ cKey ] ) TAG ( cKey ) TO ( cAlias )
    NEXT
    SET ORDER TO TAG ( cTag )
  ENDIF
  GO TOP

  RETURN .T.

```

*Listing 18: find\_fakt\_za\_period implementacija (2/3)*

```

STATIC FUNCTION use_sql_fakt_order( hParams )
    LOCAL cOrder := ""

    IF hb_HHasKey( hParams, "order_by" )
        cOrder += " ORDER BY " + hParams[ "order_by" ]
    ELSE
        cOrder += " ORDER BY idfirma,idtipdok,brdok"
    ENDIF

    RETURN cOrder

STATIC FUNCTION use_sql_fakt_where( hParams )

    LOCAL cWhere := ""
    LOCAL dDatOd

    IF hb_HHasKey( hParams, "idfirma" )
        cWhere := parsiraj_sql( "idfirma", hParams[ "idfirma" ] )
    ENDIF

    IF hb_HHasKey( hParams, "idtipdok" )
        cWhere += iif( Empty( cWhere ), "", " AND " ) + parsiraj_sql( "idtipdok", hParams[ "idvn" ] )
    ENDIF

    IF hb_HHasKey( hParams, "brdok" )
        cWhere += iif( Empty( cWhere ), "", " AND " ) + parsiraj_sql( "brdok", hParams[ "brdok" ] )
    ENDIF

    IF hb_HHasKey( hParams, "idpartner" )
        cWhere += iif( Empty( cWhere ), "", " AND " ) +
                    parsiraj_sql( "idpartner", hParams[ "idpartner" ] )
    ENDIF

    IF hb_HHasKey( hParams, "dat_do" )
        IF !hb_HHasKey( hParams, "dat_od" )
            dDatOd := CToD( "" )
        ELSE
            dDatOd := hParams[ "dat_od" ]
        ENDIF
        cWhere += iif( Empty( cWhere ), "", " AND " ) +
                    parsiraj_sql_date_interval( "datdok", dDatOd, hParams[ "dat_do" ] )
    ENDIF

    IF hb_HHasKey( hParams, "where" )
        cWhere += iif( Empty( cWhere ), "", " AND " ) + hParams[ "where" ]
    ENDIF

    RETURN cWhere

```

*Listing 19: find\_fakt\_za\_period implementacija (3/3)*

Pregledom programskog kôda (Listinzi 17-19) uočava se da se unutar funkcije formiraju standardni SQL elementi (ORDER, WHERE), koji se prosljeđuju *use\_sql()* funkciji koja na kraju podatke dohvata standardnim SQL upitom. Na ovaj način aplikativni kôd je pretrpio minimalne promjene zamjenom izvora podataka sa DBF tabelama (FMK) na relacijsku bazu (F18).

### **7.3.3 Više-platformski pristup**

“Harbour” ima podršku za sve glavne desktop operativne sisteme. “F18” klijentska aplikacija podržava Linux i Windows klijente.

## **7.4 Upravljanje projektom**

Prethodne sekcije objašnjavaju razloge i ciljeve razvoja “F18”. One ukazuju na poteškoće u održavanju i razvoju projekta. Programski jezik “harbour”, izvorni kôd u lošem stanju, arhitektura stara više od dvadeset godina, te arhaični korisnički interfejs činili su razvoj F18 neatraktivnim.

### **7.4.1 Poslovni kontekst projekta**

Do 2014. godine svi korisnici “FMK” sa manjim bazama podataka migrirali su na “F18”. Time su glavni poslovni ciljevi “F18” ostvareni. Međutim, nestabilnost u sloju baze podataka (Ref. paragraf 7.1.2: *“Ad-hoc” rješenja, dugoročne glavobolje*) onemogućila je da se proces migracije sa “FMK” na “F18” završi do kraja. Odlazak glavnog developera krajem 2014. godine, činjenica da je projekat općenito neatraktivan za razvoj, doveli su preduzeće u dilemu: “Da li je rentabilno ulagati u ovaj projekat? Da li je bolje da se usmjerimo isključivo na sistemska rješenjima u kojima smo konkurenčni?”

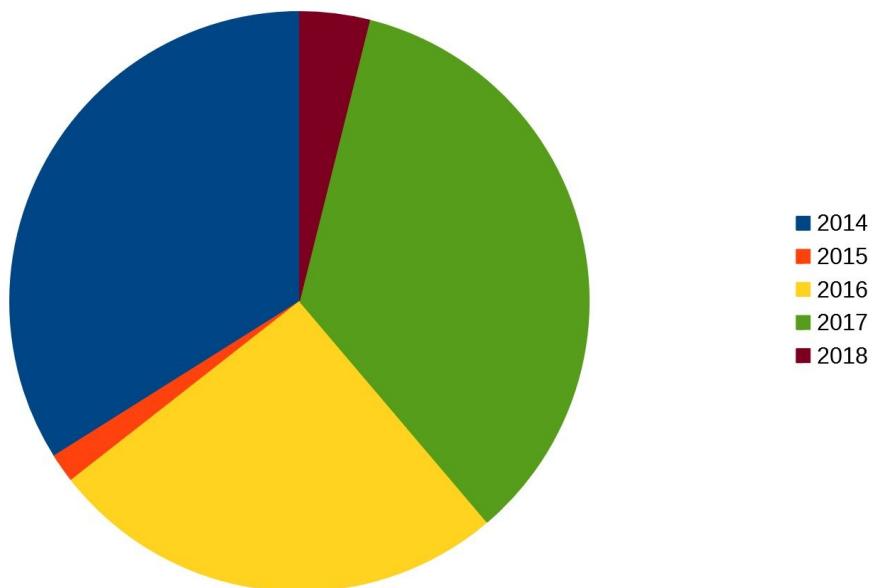
### **7.4.2 Upotrebnna vrijednost**

Zaključeno je da je većina klijenata direktno ili indirektno povezana sa preduzećem preko knjigovodstvenih aplikacija, te da postoji velika opasnost da će konkurenca nakon zamjene knjigovodstvenog softvera ugroziti pozicije preduzeća kod korisnika i po pitanju drugih rješenja i usluga. Dodatno, korisnička baza knjigovodstvenih aplikacija, u kojoj su većina klijenti preko 10 godina, govore u prilog upotrebnoj vrijednosti projekta. Softver koji funkcioniše decenijama ima bez dvojbe veliku upotrebnu vrijednost. Stoga je odlučeno da se projekat razvoja “F18” nastavi bez obzira na velike nedostatke i probleme.

## **7.5 Git istorija projekta 2014-2018**

Statistika ažuriranja u glavni git repozitorij (Slika 11) projekta je dobar pokazatelj dešavanja unutar projekta. Od ukupno preko 5000 “commit”-a, 35% otpada na 2014 godinu unutar koje se dešavao transfer projekta sa starog developera na novog, 2015 došlo je do potpune stagnacije razvoja, dok su se najbitnije aktivnosti desile u periodu 2016-2017 (>60%). U tom periodu je izvršena migracija knjigovodstva poslednjeg, ujedno i najvećeg korisnika.

```
$ git rev-list --count --since="Jan 1 2014" --before="Sep 1 2018" --all => 5024  
$ git rev-list --count --since="Jan 1 2015" --before="Dec 31 2015" --all => 82  
$ git rev-list --count --since="Jan 1 2016" --before="Dec 31 2016" --all => 1288  
$ git rev-list --count --since="Jan 1 2017" --before="Dec 31 2017" --all => 1753  
$ git rev-list --count --since="Jan 1 2018" --before="Sep 30 2018" --all => 200
```



*Slika 11: Statistika publikovanja u F18 git repozitorij 2014-2018 god.*

Istorija promjena izvornog kôda projekta na najbolji način oslikava stanje projekta:

- 2014 - odlazeći developer, veliki nivo aktivnosti
- 2015 - projekat neaktivan; dileme oko nastavka projekta
- 2016–2017 – intenzivan razvoj, migracija najvećeg klijenta na F18
- 2018 - period stabilizacije

## **7.6 Glavni ciljevi razvoja “F18” u periodu 2016-2018**

Nakon odluke da se razvoj nastavi, započela je nova etapa razvoja. Otklanjanje ranije dijagnosticiranih problema postavljeno je za primarne ciljeve:

### **7.6.1 Čitljivost izvornog kôda (eng. source code readability)**

- izbrisati komentare koji su zastarjeli ili ne sadrže nikakve korisne informacije
- preimenovati imena funkcija i varijabli da njihova imena ukazuju na namjere programskog kôda
  - npr. `FUNCTION ost_rz()` -> `FUNCTION fin_rucno_zatvaranje_otvorenih_stavki()`
- Eliminisati dijelove koji se više ne koriste, kao što su:
  - funkcije rađene specifično za klijente koji više nisu aktivni
  - zakonska rješenja koja više nisu aktuelna (npr. Obračun poreza na promet – zakonska regulativa prije stupanja zakona o PDV-u)

### **7.6.2 Promjene na sloju pristupa podataka (eng. data layer)**

- Komponentu “semafori” (sinhronizacija DBF<->SQL) u potpunosti zamijeniti sa novim rješenjem
- Performanse i jednostavnost programerskog interfejsa su prioritet
- Gornji zahtjevi neće omogućiti potpunu kompatibilnost sa FMK aplikativnim kôdom (sa “semaforima” je napravljena ta greška), ali je svakako bitno obezbjediti da potrebne prilagodbe aplikativnog kôda budu minimalne
- U roku od 6 mjeseci na novi način pristupa podacima prebaciti programske module (FIN, KALK, FAKT)

### **7.6.3 Realizacija programskih zahtjeva korisnika**

- U toku 2015 godine došlo je do prekida razvoja. Bilo je potrebno realizovati zahtjeve koji su ostali aktuelni.
- Krajem 2015 godine ugovorena je migracija posljednjeg korisnika sa “FMK” na “F18”. Proces migracije je sa sobom sadržavao niz zahtjeva na programsku nadogradnju.

## 7.7 Pregled F18, korisnički nivo

Osnovni meni aplikacije – odabir programskog modula (Slika 12):

```
Tekuća baza: f18test_2018 / db ver: 0.0.28 / nivo log: 9
Korisnik: hernad gr: [] VER: 3.1.214
-----
1. FIN # finansijsko poslovanje
2. KALK # robno-materijalno poslovanje
3. FAKT # fakturisanje
4. ePDV # elektronska evidencija PDV-a
5. LD # obračun plata
6. OS/SII# osnovna sredstva i sitan inventar
7. VIRM # virmanni
8. POS # maloprodajna kasa
-----
S. promjena sezone
B. backup podataka
U. upgrade (nadogradnja) aplikacije
P. parametri aplikacije
W. pregled log-a
X. diag info
```

Slika 12: F18 meni za odabir programskog modula

## 7.8 Postavke - parametri F18

Jedna od niza formi za podešenje parametara rada aplikacije (Slika 13):

```
Odabir modula za glavni meni ***
FIN: D KALK: D FAKT: D ePDV: D LD: D VIRM: D OS/SII: D POS: D MAT: N RNAL: N

Matični podaci korisnika ***
Puno ime i prezime: [REDACTED]

Email parametri ***
email server: [REDACTED] port: 25
username: [REDACTED]
moja_email_adresa: [REDACTED] password: [REDACTED]
slati poštu na adresu: [REDACTED]
cc adrese: [REDACTED]

Parametri log-a ***
Briši stavke log tabele starije od broja dana (def. 30): 30

Backup parametri ***
Automatski backup podataka organizacije (interval dana 0 - ne radi ništa): 2
Automatski backup podataka servera (interval 0 - ne radi ništa): 0
Udaljena backup lokacija: [REDACTED]
Ping time kod backup komande: 0

Ostali parametri ***
Dužina stranice za izvještaje (def: 60): 60
BUG report na email (D/N/A/O): A Nivo logiranja (0..9) 9

Kompatibilnost ***
KALK PR: N PTXT: D F18 LO (D/N/O): D F18 updates (D/N): D F18 verzija: 3 - std / S
```

Slika 13: Parametri F18

Pored prikazane forme za podešenje zajedničkih parametara, svaki od programskega modula ima svoj set parametara. Sveukupno, aplikacija sadrži 600-700 različitih parametara.

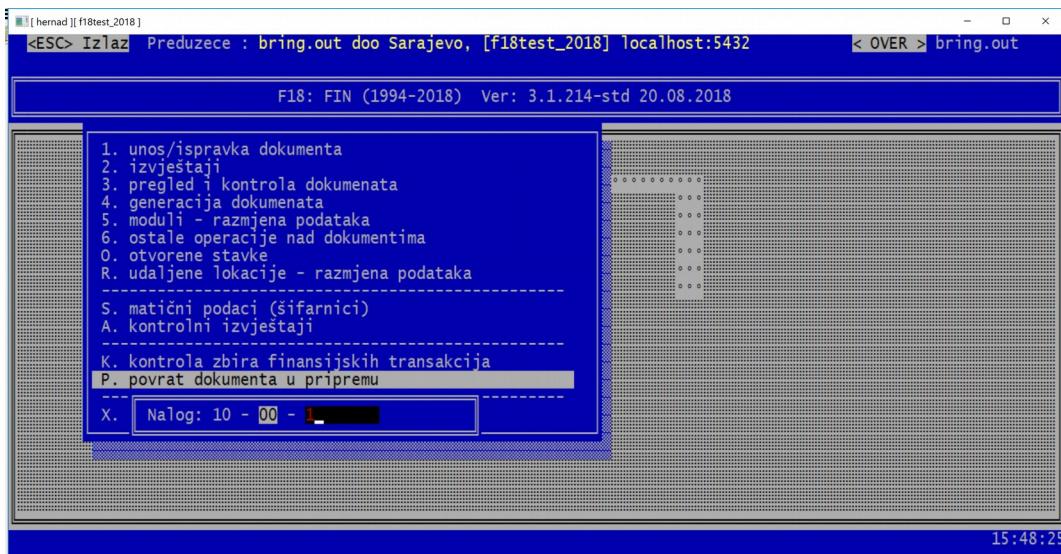
## 7.9 FIN - Finansijsko poslovanje

Osnovni programski modul je modul za praćenje finansijskog poslovanja. Obezbeđuje unos i ažuriranje finansijskih dokumenata (Slike 14 i 15), te standardne finansijske izvještaje (kartice analitičkih konta, kupaca, dobavljača, različite specifikacije, bruto bilans)

The screenshot shows a PDF document titled "LISTA FINANSIJSKIH NALOGA NA DAN: 20.08.18". The header also includes "(c) bring.out" and "Str: 000001". Below the header, it says "Preduzece : bring.out doo Sarajevo". The main content is a table with columns: R.br, FIR, V, BR, DAT, DUGUJE, POTRAŽUJE, OP., MA, N, NAL, NAL, KM, KM. The table lists 15 transactions with various values.

*	R.br	*FIR*	V	* BR	* DAT	* DUGUJE	* POTRAŽUJE	* OP.	*	*	MA	N	NAL	* NAL	* KM	* KM	*	*
1	10	00	00000001	28.02.18		140688.72	140688.72											
2	10	14	00000001	09.02.18		1390.98	1390.98											
3	10	14	00000002	09.02.18		154.71	154.71											
4	10	14	00000003	28.02.18		1248.00	1248.00											
5	10	B2	00000001	28.02.18		692.31	692.31											
6	10	B3	00000001	28.02.18		776.07	776.07											
7	10	I7	00000001	28.02.18		83.75	83.75											
8	10	IB	00000001	28.02.18		29532.42	29532.42											
9	10	IB	00000002	28.02.18		12697.24	12697.24											
10	10	IB	00000003	23.02.18		1157.53	1157.53											
11	10	IB	00000004	28.02.18		5667.70	5667.70											
12	10	IF	00000001	08.02.18		4586.86	4586.86											
13	10	IF	00000002	28.02.18		16595.28	16595.28											
14	10	IF	00000003	28.02.18		2908.16	2908.16											
15	10	OK	00000001	28.02.18		0.03	0.03											

Slika 14: Pregled ažuriranih finansijskih dokumenata (nalogi)



Slika 15: FIN - Povrat dokumenta u pripremu

Nalog nakon povrata dokument više ne ulazi u izvještaje. Premješta se u tabelu pripreme (Slika 16), gdje se može korigovati ili bezpovratno izbrisati (Slika 17).

F.	VN Br.	R.br	Konto	Partner	Br.veze	Datum	D/P	Iznos KM	Iznos EUR
10	00 00000001	1	0130		PS	01.01.18	1	5995.42	3069.64
10	00 00000001	2	0195		PS	01.01.18	2	1723.68	882.53
10	00 00000001	3	0220		PS	01.01.18	2	27979.53	14325.52
10	00 00000001	4	0221		PS	01.01.18	1	20028.45	10254.57
10	00 00000001	5	0222		PS	01.01.18	1	57981.98	29686.76
10	00 00000001	6	0223		PS	01.01.18	1	1421.30	727.70
10	00 00000001	7	02237		PS	01.01.18	1	564.02	288.78
10	00 00000001	8	0291		PS	01.01.18	2	20028.45	10254.57
10	00 00000001	9	0292		PS	01.01.18	2	15104.50	7733.51
10	00 00000001	10	0293		PS	01.01.18	2	600.33	307.37
10	00 00000001	11	1030		PS	01.01.18	1	14336.61	7340.33

<c+N> Nova stavka  
<c+A> Ispravka stavki  
<c+F9> Briši sve  
<a+T> Briši po uslovu  
FIN Priprema

<ENT> Ispravka  
<+P> Štampa naloga  
<F5> Kontrola zbirka  
<B> odredi broj dokumenta

<c+T> Briši stavku  
<a+A> Ažuriranje  
<a+F5> Pr.dat  
<F9> sredi Rbr.

<P> Povrat naloga  
<X> Ažur.bez štampe  
<a+B> Blagajna  
<F10> Ostale opcije

15:48:36

Slika 16: FIN - Tabela pripreme

Firma: 10 - bring.out doo Sarajevo

NALOG: Vrsta: 00 POČETNO STANJE Broj: 00000001

Redni broj stavke naloga: 4

DOKUMENT:  
Vezni broj: PS Datum: 01.01.18 Valuta: . .

Opis: POČETNO STANJE

Konto : 0221 TRANSPORTNA SREDSTVA  
Partner:

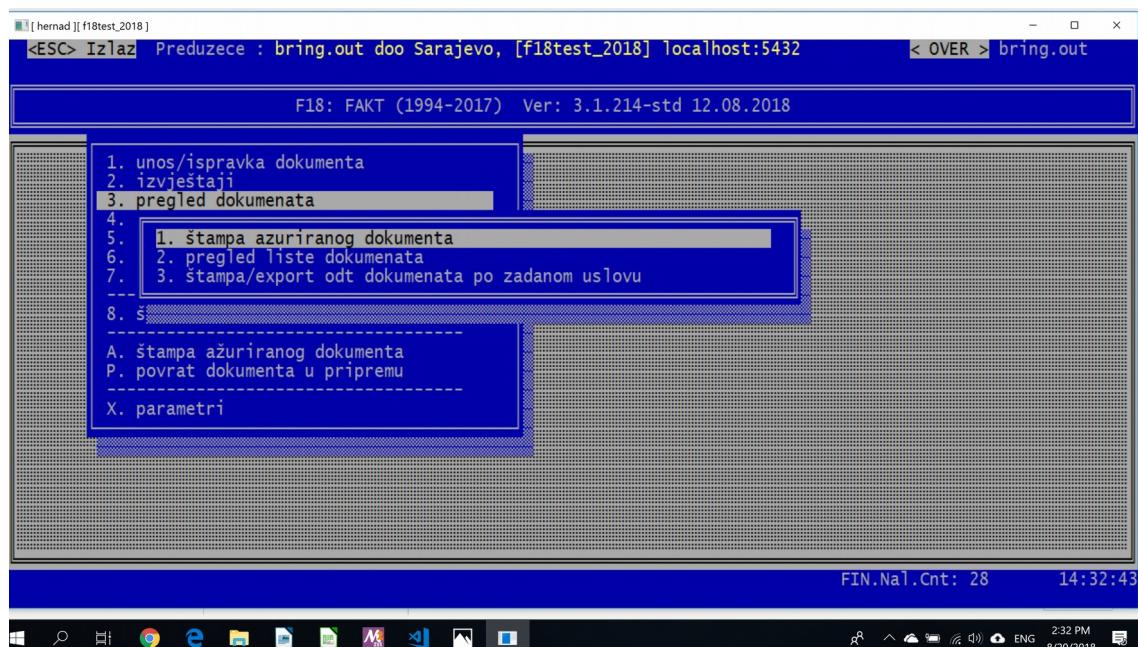
Duguje/Potražuje (1/2): 1 DUGUJE KM 20028.45 <(Alt-o) Otvorene stavke>  
DUGUJE EUR 10254.57

15:48:45

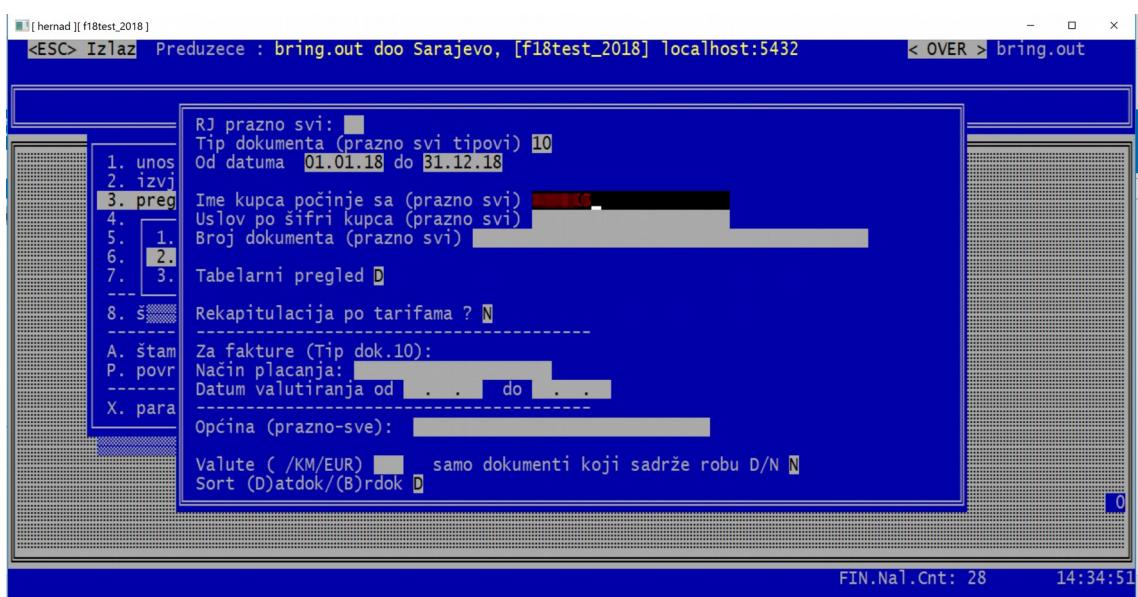
Slika 17: FIN - Stavka naloga u tabeli pripreme

## 7.10 FAKT - modul za fakturisanje

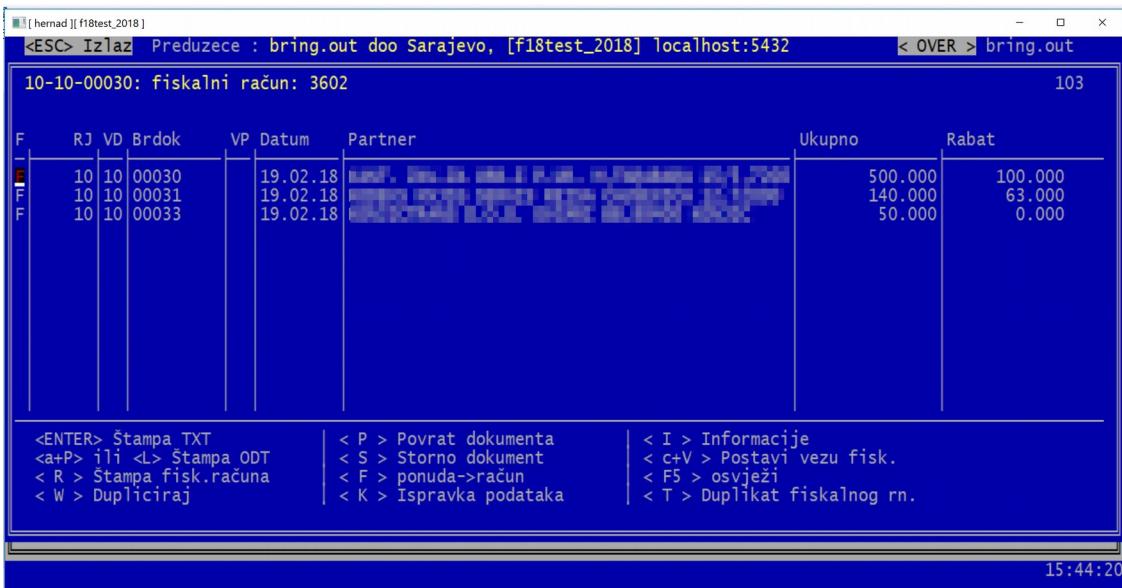
Komercijalni poslovi se obavljaju kroz ovaj modul. U njemu se evidentiraju ponude i izdaju fakture. Prikaz operacija koje korisniku omogućavaju pregled i štampanje izlaznih faktura (Slike 18-21):



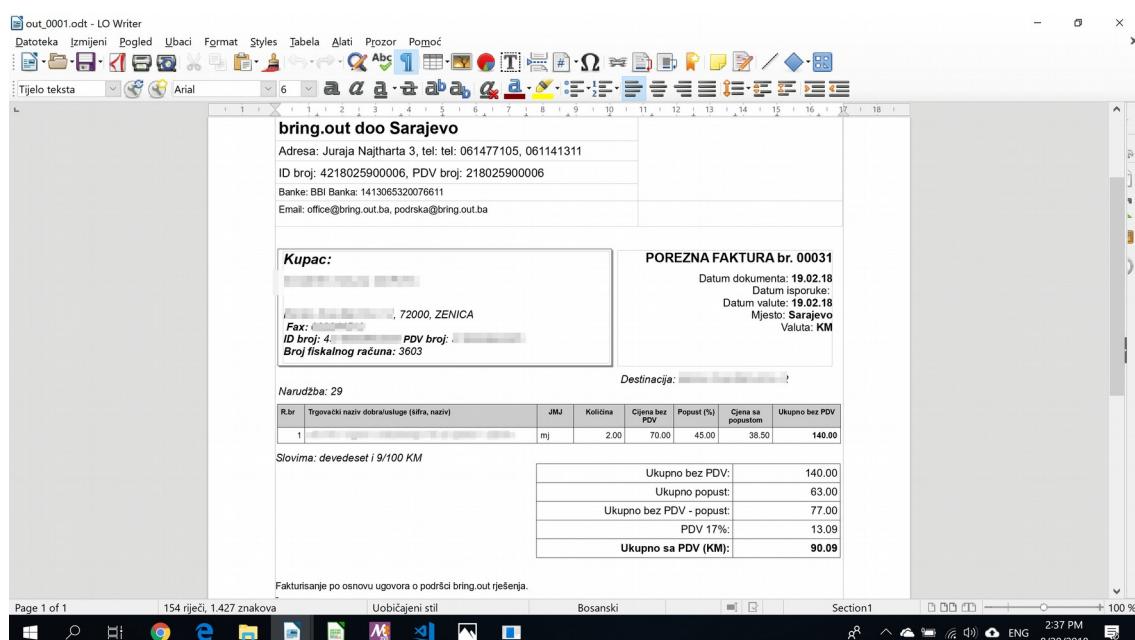
Slika 18: FAKT - Osnovni meni, podmeni "pregled dokumenata"



Slika 19: FAKT - Forma za unos podataka: zadavanje željene vrste dokumenata i partnera (tip 10-fakture)



Slika 20: Tabelarni pregled FAKT dokumenata



Slika 21: FAKT - pregled fakture u LibreOffice

## **7.11 Ostali programski moduli**

Ostali programski moduli su:

- KALK - Kalkulacije, program za robno/materijalno poslovanje i praćenje proizvodnje
- LD - Obračun plata
- ePDV - evidencija KUF, KIF, obračun PDV-a
- POS - trgovačka kasa na malo sa fiskalnim funkcijama
- OS - evidencija i obračun amortizacije stalnih sredstava i sitnog inventara
- VIRM - unos i štampanje virmana

## 8 Razvoj projekta “F18” u periodu 2014-2018

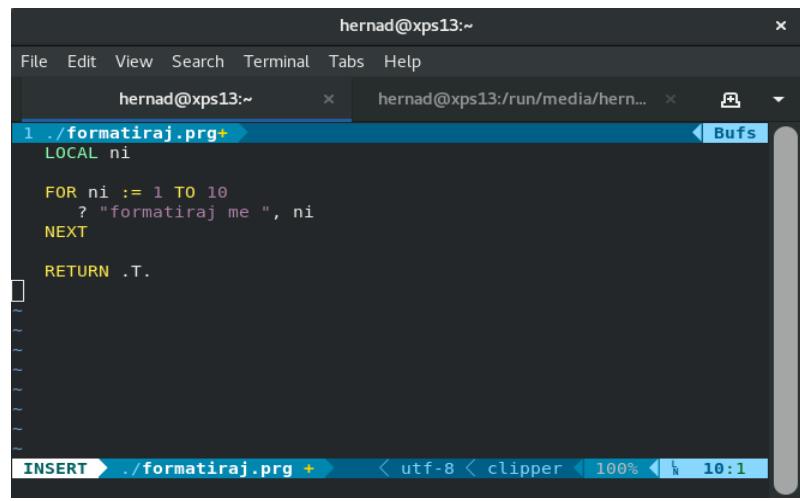
### 8.1 Uvod

U ovom dijelu rada navedene su glavne aktivnosti i rezultati razvoja projekta “F18” u periodu 2014.-2018. Prikazaćemo najvažnije aktivnosti, te na kraju analizirati realizovano u kontekstu agilnih metoda i praksi.

### 8.2 “Atom” editor i podrška za “harbour”

Programerski editor je osnovni alat developera. Do 2014. se za razvoj koristio isključivo “vim” editor<sup>32</sup> i set unix alata (grep, sed, awk) [SEDAWK]. Najveći problem je bila minimalna podrška editora “harbour” programskom jeziku. S obzirom na nisku popularnost “harbour”-a, moderni programerski editori nisu nudili nikakvu podršku. S druge strane, pojavila se nova generacija programerskih editora, koja je u osnovnoj arhitekturi značajno dodataka za nove programske jezike. “Atom”<sup>33</sup> programerski editor identificiran je kao najbolja platforma za razvoj ekstenzija koje će omogućiti veću produktivnosti “harbour” programeru. Rezultat tog rada su tri ekstenzije za atom editor:

- Podrška harbour sintaksi:
  - <https://github.com/hernad/atom-language-harbour>
- Formatiranje harbour koda (hbformat):
  - <https://github.com/hernad/atom-harbour-plus>
- Provjera sintakse i kvaliteta harbour kôda uz pomoć harbour kompjlera:
  - <https://github.com/AtomLinter/linter-harbour>



Slika 22: “vim” editor

Atom je OSS i multi-platformski softver. Izgrađen je na web tehnologijama.

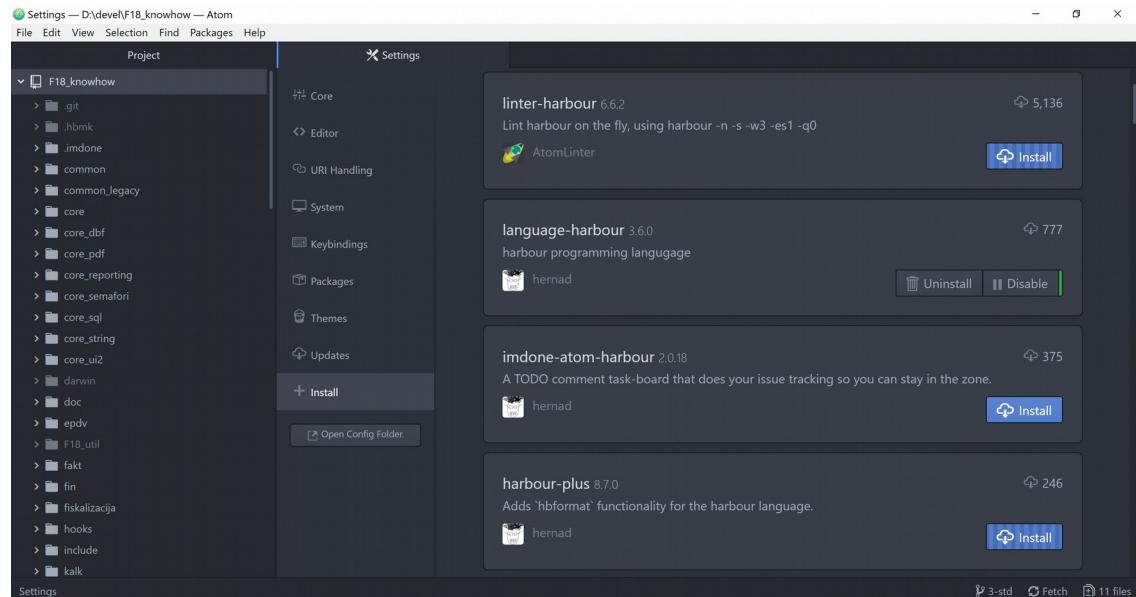
Njegovo jezgro čini “elektron” <https://electronjs.org>. Bazna komponenta “elektrona” je “google chromium” browser <https://www.chromium.org/Home>

Zanimljivost 2: Arhitektura “atom” editora

32 <https://www.vim.org>

33 <https://atom.io>

Ekstenzije su OSS sotver izdane pod MIT<sup>34</sup> licencom. Napisane su u “coffeescript”<sup>35</sup> programskom jeziku. Autor ovog rada je autor ekstenzija. Dostupne su svakom korisniku “atom” editora kroz opciju “install” (Slika 23):



Slika 23: "Atom" ekstenzije za harbour (harbour-linter, harbour-language, harbour-plus)

### 8.2.1 Atom “language-harbour” ekstenzija

Ova ekstenzija obezbjeđuje prepoznavanje sintakse programskog jezika (Slika 24) :

The screenshot shows the Atom code editor with the file 'core\_0.prg' open. The code is written in Harbour language. The editor interface includes a 'Project' sidebar on the left and various status icons at the bottom.

```

Project — D:\devel\F18_knowhow — Atom
File Edit View Selection Find Packages Help
Project
F18_knowhow
  .git
  .hbmk
  .imdone
    config.json
  common
  common_legacy
  core
    clipboard.prg
    cls_dok_attr.prg
  core_0.prg
  core_colors.prg
  core_hb_util.prg
  core_os.prg
  download_template.prg
  editor.prg
  f18_editor.prg
  f18_update.prg
  f18_ver.prg
  java_download.prg
  jodeports_download.prg
  LO_download.prg
  pdf_download.prg
  psql_download.prg
core/core_0.prg  1:1
Settings
core_0.prg
  * it is licensed to you under the COMMON PUBLIC ATTRIBUTION LICENSE
  * version 1.0, the full text of which (including FMK specific Exhibits)
  * is available in the file LICENSE_CPALE_BRING.out_knowhow.md located at the
  * root directory of this source code archive.
  * By using this software, you agree to be bound by its terms.
  */
11
12 #include "f18.ch"
13
14
15 FUNCTION harbour_init()
16
17   rddSetDefault( RDDENGINE )
18   Set( _SET_AUTOPEN, .F. )
19
20   SET CENTURY OFF
21   SET EPOCH TO 1980 // 81 - 1981, 79-2079
22   SET DATE TO GERMAN
23
24   f18_init_threads()
25
26   Set( _SET_OSCODEPAGE, hb_cdpOS() )
27
28 // ? SET( _SET_OSCODEPAGE )
29
LF  UTF-8  harbour  3-std  Fetch  11 files
Windows Taskbar: File Explorer, Edge, Task View, Taskbar Icons, Taskbar Buttons, Taskbar Notifications, Taskbar Clock, Taskbar Language, Taskbar Volume, Taskbar Network, Taskbar Battery, Taskbar Date

```

Slika 24: Podrška sintaksi “harbour” programskog jezika

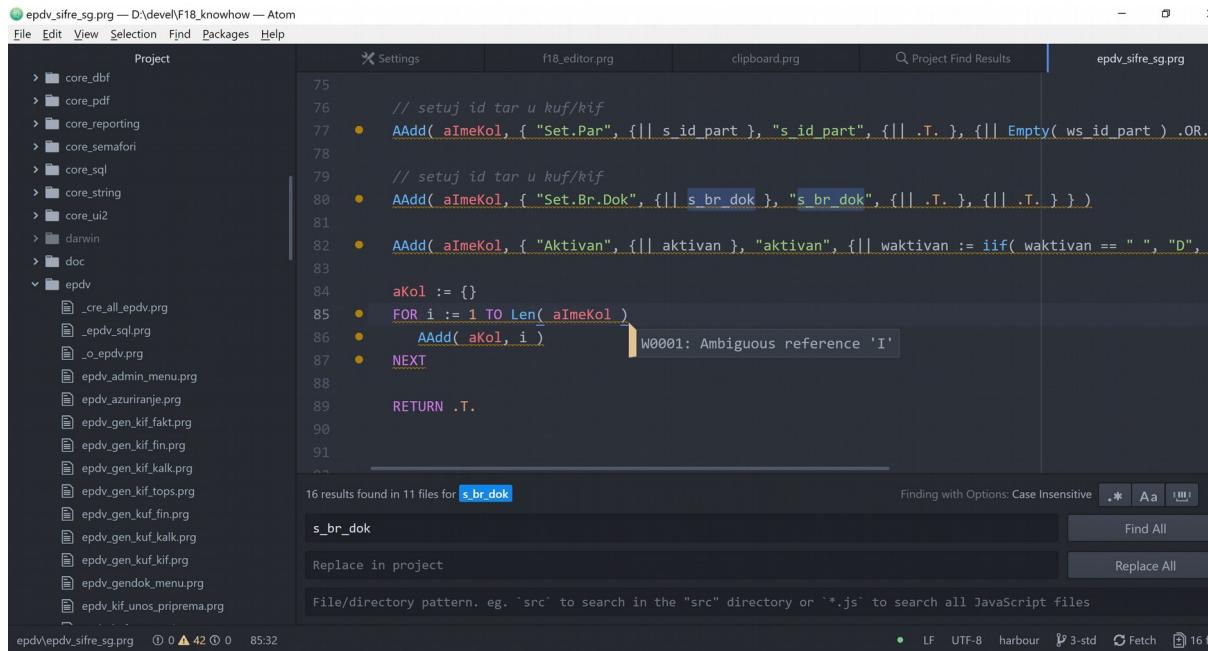
34 <https://opensource.org/licenses/MIT>

35 <https://coffeescript.org/>

## 8.2.2 Atom “linter-harbour” ekstenzija

Ova ekstenzija je najviše doprinijela produktivnosti developer-a.

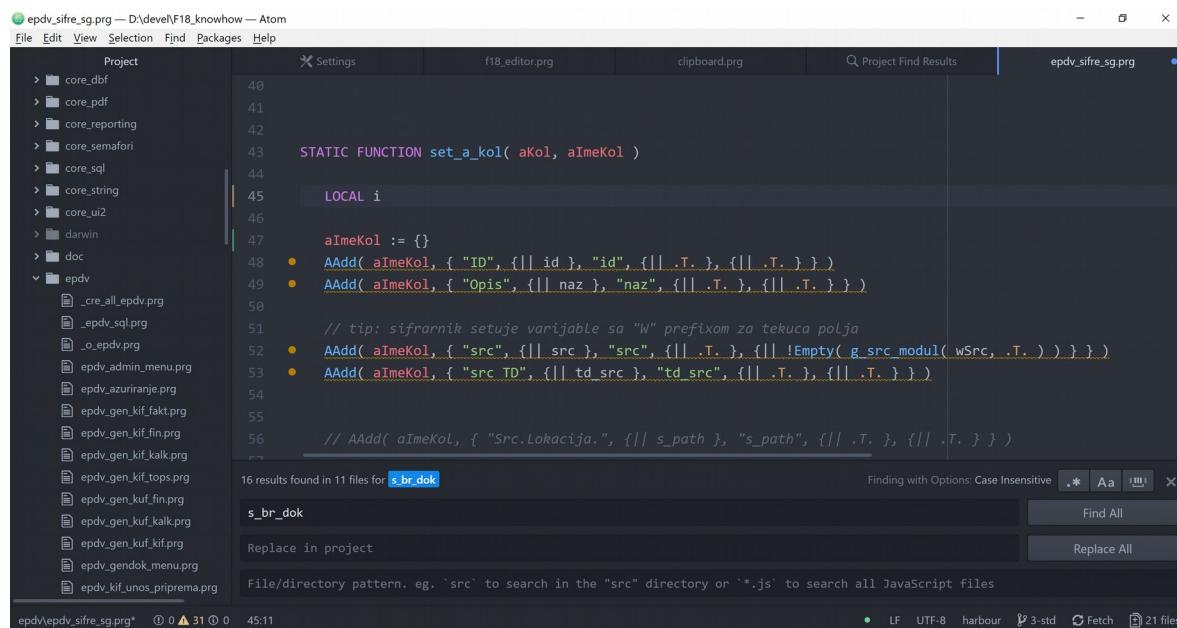
“Linter” ekstenzija koji u toku procesa ispravke programskog fajla prikazuje upozorenja (eng. warning, smeđa boja) i greške (eng. error crvena boja). Ekstenzija time omogućava efikasan refaktoring programskog kôda.



```
epdv_sifre_sg.prg — D:\devel\F18_knowhow — Atom
File Edit View Selection Find Packages Help
Project Settings f18_editor.prg clipboard.prg Project Find Results epdv_sifre_sg.prg
75
76    // setuj id tar u kuf/kif
77 ● AAdd( aImeKol, { "Set.Pan", {|| s_id_part }, "s_id_part", {|| .T. }, {|| Empty( ws_id_part ) } .OR.
78
79    // setuj id tar u kuf/kif
80 ● AAdd( aImeKol, { "Set.Br.Dok", {|| s_br_dok }, "s_br_dok", {|| .T. }, {|| .T. } } )
81
82 ● AAdd( aImeKol, { "Aktivan", {|| aktivan }, "aktivan", {|| waktivan := iif( waktivan == "", "D",
83
84     aKol := {}
85 ● FOR i := 1 TO Len( aImeKol )
86     AAdd( aKol, i )
87 ● NEXT
88
89     RETURN .T.
90
91
16 results found in 11 files for s_br_dok
s_br_dok
Find All
Replace in project
Replace All
File/directory pattern. eg. 'src' to search in the "src" directory or '*.js' to search all JavaScript files
epdv\epdv.sifre_sg.prg ① 0 ▲ 42 ① 0 85:32
LF UTF-8 harbour 3-std Fetch 16 files
```

Slika 25: ”harbour” linter daje sugestije za ispravke koda

Na slici 25 se vidi način na koji “linter” djeluje. On daje sugestiju da privatnu varijablu “i” treba deklarisati kao lokalnu. Na dnu ekrana se prikazana rekapitulacija upozorenja i grešaka za programski fajl na kome radimo ([42 upozorenja](#), [0 grešaka](#)). Rezultate ispravki prikazuju slike 26 i 27:



```
epdv_sifre_sg.prg — D:\devel\F18_knowhow — Atom
File Edit View Selection Find Packages Help
Project Settings f18_editor.prg clipboard.prg Project Find Results epdv_sifre_sg.prg
40
41
42
43 STATIC FUNCTION set_a_kol( aKol, aImeKol )
44
45 LOCAL i
46
47 aImeKol := {}
48 ● AAdd( aImeKol, { "ID", {|| id }, "id", {|| .T. }, {|| .T. } } )
49 ● AAdd( aImeKol, { "Opis", {|| naz }, "naz", {|| .T. }, {|| .T. } } )
50
51 // tip: sifrarnik setuje varijable sa "W" prefixom za tekuća polja
52 ● AAdd( aImeKol, { "src", {|| src }, "src", {|| .T. }, {|| !Empty( g_src_modul( wSrc, .T. ) ) } } )
53 ● AAdd( aImeKol, { "src TD", {|| td_src }, "td_src", {|| .T. }, {|| .T. } } )
54
55
56 // AAdd( aImeKol, { "Src.Lokacija.", {|| s_path }, "s_path", {|| .T. }, {|| .T. } } )
16 results found in 11 files for s_br_dok
s_br_dok
Find All
Replace in project
Replace All
File/directory pattern. eg. 'src' to search in the "src" directory or '*.js' to search all JavaScript files
epdv\epdv.sifre_sg.prg* ① 0 ▲ 31 ① 0 45:11
LF UTF-8 harbour 3-std Fetch 21 files
```

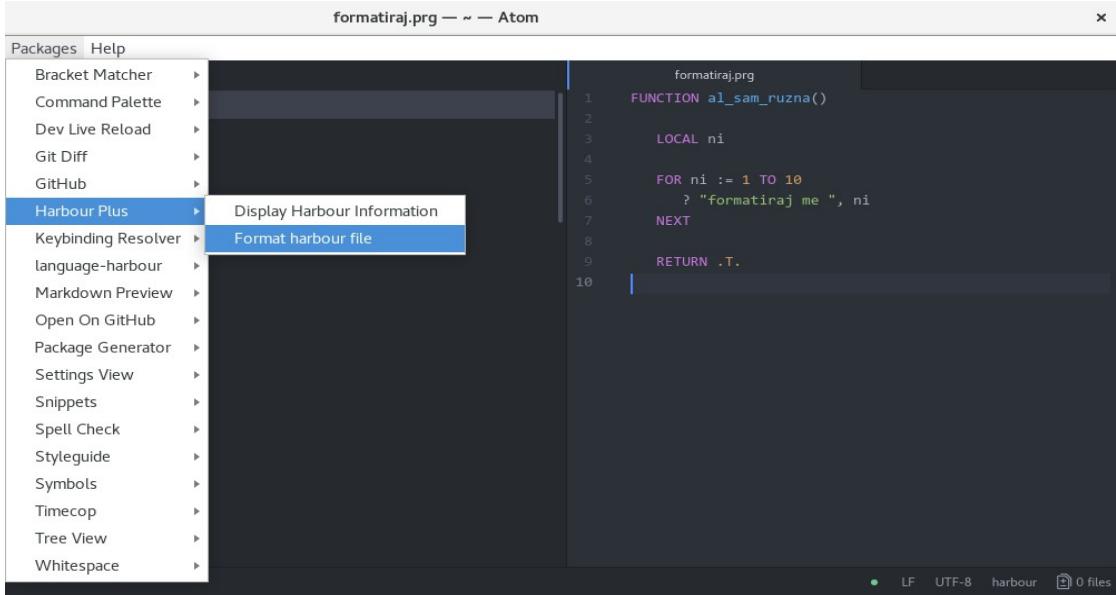
Slika 26: ispravka kôda prema sugestiji “harbour” lintera

The screenshot shows the Atom code editor with a Harbour script file open. The code contains several instances of the identifier `s_br_dok`. A tooltip or status bar at the bottom indicates "16 results found in 11 files for `s_br_dok`". The editor interface includes a sidebar with a project tree, a search bar, and various status indicators like file count and encoding.

```

epdv_sifre_sg.prg — D:\devel\F18_knowhow — Atom
File Edit View Selection Find Packages Help
Project
  core_dbf
  core_pdf
  core_reporting
  core_semafori
  core_sql
  core_string
  core_ui2
  darwin
  doc
  epdv
    _cre_all.epdv.prg
    _epdv_sql.prg
    _o_epdv.prg
    epdv_admin_menu.prg
    epdv_azuriranje.prg
    epdv_gen_kif.fakt.prg
    epdv_gen_kif.fin.prg
    epdv_gen_kif.kalk.prg
    epdv_gen_kif.tops.prg
    epdv_gen_kuf_fin.prg
    epdv_gen_kuf_kalk.prg
    epdv_gen_kuf_kif.prg
    epdv_gendok_menu.prg
    epdv_kif_unos_sprema.prg
  f18_editor.prg
  clipboard.prg
  Project Find Results
  epdv_sifre_sg.prg
  Settings
  f18_editor.prg
  clipboard.prg
  epdv_sifre_sg.prg
  epdv_sifre_sg.prg
  78 • AAdd( aImeKol, { "Set.Par", {|| s_id_part }, "s_id_part", {|| .T. }, {|| Empty( ws_id_part ) } .OR. p_partne
  79
  80 // setuj id tar u kuf/kif
  81 • AAdd( aImeKol, { "Set.Br.Dok", {|| s_br_dok }, "s_br_dok", {|| .T. }, {|| .T. } } )
  82
  83 • AAdd( aImeKol, { "Aktivan", {|| aktivan }, "aktivan", {|| waktivan := iif( waktivan == " ", "D", waktivan
  84
  85 akol := {}
  86 FOR i := 1 TO Len( aImeKol )
  87   AAdd( akol, i )
  88 NEXT
  89
  90 RETURN .T.
  91
  92
  93 FUNCTION info_partner()
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1018
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1028
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1038
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1048
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1058
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1068
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1078
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1088
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1098
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1108
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1117
  1118
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1127
  1128
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1137
  1138
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1146
  1147
  1148
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1157
  1158
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1166
  1167
  1168
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1176
  1177
  1178
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1187
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1195
  1196
  1197
  1198
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1206
  1207
  1208
  1209
  1210
  1211
  1212
  1213
  1214
  1215
  1216
  1217
  1218
  1219
  1220
  1221
  1222
  1223
  1224
  1225
  1226
  1227
  1228
  1229
  1230
  1231
  1232
  1233
  1234
  1235
  1236
  1237
  1238
  1239
  1240
  1241
  1242
  1243
  1244
  1245
  1246
  1247
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1255
  1256
  1257
  1258
  1259
  1260
  1261
  1262
  1263
  1264
  1265
  1266
  1267
  1268
  1269
  1270
  1271
  1272
  1273
  1274
  1275
  1276
  1277
  1278
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1286
  1287
  1288
  1289
  1290
  1291
  1292
  1293
  1294
  1295
  1296
  1297
  1298
  1299
  1300
  1301
  1302
  1303
  1304
  1305
  1306
  1307
  1308
  1309
  1310
  1311
  1312
  1313
  1314
  1315
  1316
  1317
  1318
  1319
  1320
  1321
  1322
  1323
  1324
  1325
  1326
  1327
  1328
  1329
  1330
  1331
  1332
  1333
  1334
  1335
  1336
  1337
  1338
  1339
  1340
  1341
  1342
  1343
  1344
  1345
  1346
  1347
  1348
  1349
  1350
  1351
  1352
  1353
  1354
  1355
  1356
  1357
  1358
  1359
  1360
  1361
  1362
  1363
  1364
  1365
  1366
  1367
  1368
  1369
  1370
  1371
  1372
  1373
  1374
  1375
  1376
  1377
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1385
  1386
  1387
  1388
  1389
  1390
  1391
  1392
  1393
  1394
  1395
  1396
  1397
  1398
  1399
  1400
  1401
  1402
  1403
  1404
  1405
  1406
  1407
  1408
  1409
  1410
  1411
  1412
  1413
  1414
  1415
  1416
  1417
  1418
  1419
  1420
  1421
  1422
  1423
  1424
  1425
  1426
  1427
  1428
  1429
  1430
  1431
  1432
  1433
  1434
  1435
  1436
  1437
  1438
  1439
  1440
  1441
  1442
  1443
  1444
  1445
  1446
  1447
  1448
  1449
  1450
  1451
  1452
  1453
  1454
  1455
  1456
  1457
  1458
  1459
  1460
  1461
  1462
  1463
  1464
  1465
  1466
  1467
  1468
  1469
  1470
  1471
  1472
  1473
  1474
  1475
  1476
  1477
  1478
  1479
  1480
  1481
  1482
  1483
  1484
  1485
  1486
  1487
  1488
  1489
  1490
  1491
  1492
  1493
  1494
  1495
  1496
  1497
  1498
  1499
  1500
  1501
  1502
  1503
  1504
  1505
  1506
  1507
  1508
  1509
  1510
  1511
  1512
  1513
  1514
  1515
  1516
  1517
  1518
  1519
  1520
  1521
  1522
  1523
  1524
  1525
  1526
  1527
  1528
  1529
  1530
  1531
  1532
  1533
  1534
  1535
  1536
  1537
  1538
  1539
  1540
  1541
  1542
  1543
  1544
  1545
  1546
  1547
  1548
  1549
  1550
  1551
  1552
  1553
  1554
  1555
  1556
  1557
  1558
  1559
  1560
  1561
  1562
  1563
  1564
  1565
  1566
  1567
  1568
  1569
  1570
  1571
  1572
  1573
  1574
  1575
  1576
  1577
  1578
  1579
  1580
  1581
  1582
  1583
  1584
  1585
  1586
  1587
  1588
  1589
  1590
  1591
  1592
  1593
  1594
  1595
  1596
  1597
  1598
  1599
  1600
  1601
  1602
  1603
  1604
  1605
  1606
  1607
  1608
  1609
  1610
  1611
  1612
  1613
  1614
  1615
  1616
  1617
  1618
  1619
  1620
  1621
  1622
  1623
  1624
  1625
  1626
  1627
  1628
  1629
  1630
  1631
  1632
  1633
  1634
  1635
  1636
  1637
  1638
  1639
  1640
  1641
  1642
  1643
  1644
  1645
  1646
  1647
  1648
  1649
  1650
  1651
  1652
  1653
  1654
  1655
  1656
  1657
  1658
  1659
  1660
  1661
  1662
  1663
  1664
  1665
  1666
  1667
  1668
  1669
  1670
  1671
  1672
  1673
  1674
  1675
  1676
  1677
  1678
  1679
  1680
  1681
  1682
  1683
  1684
  1685
  1686
  1687
  1688
  1689
  1690
  1691
  1692
  1693
  1694
  1695
  1696
  1697
  1698
  1699
  1700
  1701
  1702
  1703
  1704
  1705
  1706
  1707
  1708
  1709
  1710
  1711
  1712
  1713
  1714
  1715
  1716
  1717
  1718
  1719
  1720
  1721
  1722
  1723
  1724
  1725
```

Formatiranjem dobijamo potrebnu preglednost - čitljivost izvornog kôda (Slika 29):



The screenshot shows the Atom editor interface with a dark theme. A context menu is open from the 'Harbour Plus' package, specifically the 'Format harbour file' option, which is highlighted with a blue background. The main editor area contains a snippet of Harbour language code:

```
formatiraj.prg — ~ — Atom
formatiraj.prg
1 FUNCTION al_sam_ruzna()
2
3 LOCAL ni
4
5 FOR ni := 1 TO 10
6 ? "formatiraj me ", ni
7 NEXT
8
9 RETURN .T.
```

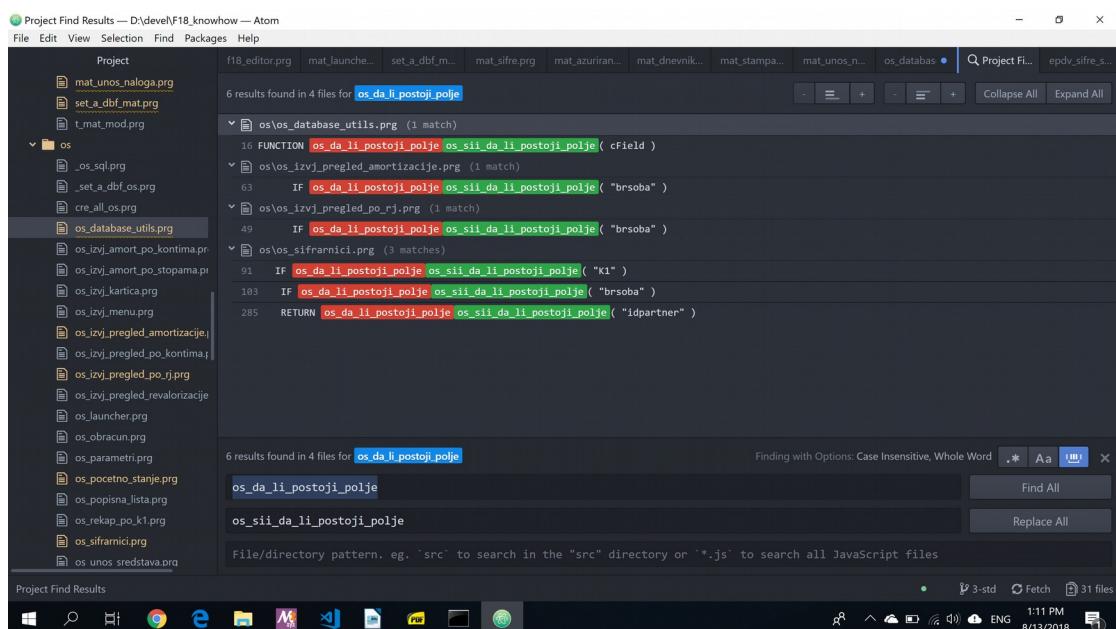
At the bottom of the editor, there are status indicators for LF, UTF-8, harbour, and 0 files.

Slika 29: "harbour-plus" ekstenzija automatski formatira programski kôd

Ranija praksa, po kojoj je svaki developer formatirao kôd na svoj način, učinila je projekat iznimno "neurednim" (teško čitljivim). Ova opcija pomogla je da se kôd brzo unificira bez puno truda, ali i grešaka koje ručno formatiranje sa sobom nosi.

### 8.2.4 Atom editor osnovne funkcije

Atom kao moderan editor ima odlično riješene "find-replace" funkcije koje podržavaju "regex" izraze<sup>36</sup>. Izmjena naziva funkcije na čitavom projektu je time postao rutinski posao (Slika 30):



Slika 30: "search/replace" funkcije u "atom" editoru

36 [https://en.wikipedia.org/wiki/Regular\\_expression](https://en.wikipedia.org/wiki/Regular_expression)

Nakon promjene treba uočiti da su svi fajlovi koji su pretrpjeli promjene promjenili boju (Slika 31):



The screenshot shows a Windows desktop environment. In the center, there is a code editor window titled "os\_database\_utils.prg" which contains Delphi-style pseudocode. The code defines a function "os\_sii\_da\_li\_postoji\_polje" that checks if a field exists in a database table. It uses LOCAL variables and IF-ELSE blocks to determine the result based on the value of "gOsSli". The code editor has syntax highlighting and line numbers. At the bottom of the code editor, there is a status bar showing file paths, line numbers, and other development tools like LF, UTF-8, harbour, Fetch, and file counts. The taskbar at the bottom of the screen shows icons for various applications including File Explorer, Microsoft Edge, and FileZilla. The system tray on the right side of the taskbar includes icons for network connection, battery level, volume, and date/time.

```
Project f18_editor.prg mat.launch... set_a_dbf.m... mat_sifre.prg mat_azuriran... mat_dnevnik... mat_stampa... mat_unos.n... os.databases Project Fi... epdv_sifre.s...
```

```
mat_unos.nalogna.prg
set_a_dbf.mat.prg
t_mat.modul.prg

os
    os_sql.prg
    _set_a_dbf_os.prg
    cre_all.os.prg
    os_database_utils.prg
    os_ivzi_amort_po_kontima.prg
    os_ivzi_amort_po_stopama.prg
    os_ivzi_kartica.prg
    os_ivzi_menu.prg
    os_ivzi_pregled_amortizacije.prg
    os_ivzi_pregled_po_kontima.prg
    os_ivzi_pregled_po_rj.prg
    os_ivzi_pregled_revalorizacije.prg
    os_launcher.prg
    os_obracun.prg
    os_parametri.prg
    os_pocetno_stanje.prg
    os_popisna_lista.prg
    os_rekap_po_k1.prg
    os_siframici.prg
    os_unos_sredstava.prg
```

```
8     * root directory of this source code archive.
9     * By using this software, you agree to be bound by its terms.
10    */
11
12    #include "f18.ch"
13
14 MEMVAR gOsSli
15
16 FUNCTION os_sii_da_li_postoji_polje( cField )
17
18    LOCAL lRet := .F.
19
20    IF gOsSli == "0"
21        IF os->( FieldPos( cField ) ) <> 0
22            lRet := .T.
23        ENDIF
24    ELSE
25        IF sii->( FieldPos( cField ) ) <> 0
26            lRet := .T.
27        ENDIF
28    ENDIF
29
30    RETURN lRet
31
```

*Slika 31: "atom" editor: identificiranje izvršenih promjena uz pomoć boja*

Masovne promjene na projektu su takođe postale jednostavan posao. Promjena sadržaja u 700 fajlova više nije dugotrajna i rizična operacija (Slika 32):

The screenshot shows the Atom code editor's 'Project Find Results' interface. The left sidebar displays the project structure of 'F18\_knowhow'. The main pane shows search results for the pattern 'Copyright (c) 1994-2011'. There are two sections of results: one for 'Copyright (c) 1994-2011' and another for 'Copyright (c) 1994-2018'. The results list various files across the project, such as 'epdv/epdv\_tarifa\_utils.prg', 'epdv/epdv\_unos\_valid.prg', and 'core/clipboard.prg'. The bottom of the interface includes a search bar, a 'File/directory pattern' input, and several status indicators.

Project Find Results — D:\dev\|F18\_knowhow — Atom

File Edit View Selection Find Packages Help

Project

672 results found in 671 files for **Copyright (c) 1994-2011**

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/epdv\_tarifa\_utils.prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/epdv\_unos\_valid.prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/epdv\_utils\_date.prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/epdv\_utils\_gen\_dok.prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/epdv\_utils\_kuf\_kif.prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/set\_a\_dbf\_epdv.prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* epdv/t\_epdv\_mod\_prg (1 match)

4 \* **Copyright (c) 1994-2011 Copyright (c) 1994-2018** by bring.out doo Sarajevo.

4 \* F18.nro (1 match)

672 results found in 671 files for **Copyright (c) 1994-2018**

Copyright (c) 1994-2011

copyright (c) 1994-2018

Finding with Options: Case Insensitive, Whole Word

Find All Replace All

File/directory pattern. e.g. 'src' to search in the 'src' directory or '\*.js' to search all JavaScript files

Project Find Results

Slika 32: "atom" editor: sugestije prije nego se naprave promjene

Uz ove ekstenzije postiglo se da developer najbitnije operacije obavlja unutar editora, bez potrebe za korištenjem eksternih alata. Složenost korištenja ranijih alata (“vim”, “grep”, “sed”, “awk”) je za većinu developera predstavljala barijeru zbog kojih gore prikazane operacije ne samo da su bile zahtjevne, nego najčešće nisu ni rađene.

### 8.2.5 Zaključak

Zahtjevnost razvoja “atom” ekstenzija učinila je da to postane zaseban projekat. Taj podprojekat je uzeo značajno vrijeme i programerske resurse nauštrb direktnog razvoja “F18”. Prva izjava u Manifestu, u kojima se navode osnovne vrijednosti agilnog razvoja, glasi: “*Osobe i njihove interakcije više od procesa i alata*”.

Da li to znači da se prevelika pažnja posvetila razvojnim alatima? Manifest ukazuje na to da akteri projekta trebaju biti prvom mjestu. Te osobe su developeri, korisnici, investitori - svi oni koji su na bilo koji način zainteresovani i uključeni u projekat. “Atom” editor i ekstenzije za “harbour” su obezbjedile produktivnost i ugodniji rad developera. Operacije koje su bile napor postale su dostupne nadohvat ruke. Razvoj “F18” je postao puno **ugodniji**. Odluka da se unaprijede razvojni alati zato ima direktno uporište u sljedećim agilnim praksama i konceptima:

1. stvaranje ugodnog okruženja za developere kroz povećanje produktivnosti
  - developer se više ne žali da je prisiljen koristiti zastarjele alate
2. poticanje razvojnog tima na inovacije i unapređenje u svim elementima djelovanja
  - stvara se svijest unutar razvojnog tima da će se cijeniti svako unapređenje razvojnog procesa, a ne samo direktno izvršenje postavljenih zadataka
3. multifunkcionalan razvojni tim
  - ako postoji potreba, član tima će se uhvatiti u koštač i sa problemima koji nisu dio njegove ekspertize (u konkretnom slučaju za realizaciju je bilo potrebno ovladati potpuno novim programskim jezikom i okruženjem)

U ovoj analizi ne smiju se izostaviti rizici koje ovakvi projekti sa sobom nose. Nove i nepoznate tehnologije uvijek sa sobom nose dodatni rizik u pogledu ishoda. Kada razvojni tim treba uraditi zadatak koji je sličan nizu prethodnih, pozitivan ishod je izvjestan. Kada se uputi u nepoznatu oblast, rizik neuspjeha je veliki. Da bi posljedice eventualnog neuspjeha bile minimalne, u projektima ove vrste posebnu važnost ima agilna praksa vremensko ograničenje realizacije (eng. **timeboxing**)<sup>37</sup>.

---

<sup>37</sup> Ref. paragraf: “Sprintovi” 3.3

## 8.3 Eliminacija programskog “otpada”

Unaprijeđeni razvojni alati su stvorili preduslove da se izvrši agresivno “čišćenje” programskog kôda projekta. Subjektivni dojam je da je na tom planu učinjeno mnogo. Međutim, rezultati ovih aktivnosti mogu se izraziti i konkretnom metrikom - brojem linija programskog kôda na početku i na kraju ravnog perioda. Podaci su izgenerirani sa “cloc”-om<sup>38</sup> (Tabele 2-5). Broj programske fajlova se smanjio neznatno (<10%), ali je broj linija programskog kôda smanjen za skoro 50%! Uklanjanje “mrtvog” i dupliranog kôda je dalo značajne rezultate. Uočljivo je “čišćenje” SQL baze: smanjenje sa 73000 na 3000 linija SQL kôda!

### 8.3.1 Stanje 2014.

github.com/AlDanial/cloc v 1.72				
Language	files	blank	comment	code
xBase	821	85575	39690	226491
SQL	2	26820	18864	73769
xBase Header	26	966	320	3759
Bourne Shell	13	137	26	236
DOS Batch	4	78	16	146
Bourne Again Shell	3	63	11	109
SUM:	869	113639	58927	304510

Tabela 2: Broj linija programskog kôda po programskim jezicima (2014)

File	files	blank	comment	code
F18_SQL.cloc	2	26820	18864	73769
F18_kalk.cloc	168	16208	6743	50954
F18_common.cloc	218	20093	11755	47140
F18_ld.cloc	70	8971	3552	26724
F18_fin.cloc	73	9112	3140	26493
F18_fakt.cloc	64	7665	3254	20263
F18_rnal.cloc	67	10024	5040	19607
F18_pos.cloc	66	5215	2644	15100
F18_epdv.cloc	39	2765	1638	5783
F18_mat.cloc	24	1918	591	5651
F18_kadev.cloc	21	1919	636	5537
F18_os.cloc	23	1528	560	4380
F18_virm.cloc	15	1125	457	2620
F18_scripts.cloc	19	276	53	489
SUM:	869	113639	58927	304510

Tabela 3: Broj linija programskog kôda po programskim modulima (2014)

38 <https://github.com/AlDanial/cloc>

### 8.3.2 Stanje 2018.

github.com/AlDanial/cloc v 1.72				
Language	files	blank	comment	code
xBase	740	70869	23820	171308
SQL	4	1808	1540	2971
xBase Header	18	698	571	2577
Bourne Shell	32	262	67	522
Markdown	4	194	0	459
DOS Batch	6	100	39	171
Bourne Again Shell	3	63	11	109
SUM:	807	73994	26048	178117

Tabela 4: Broj linija programskog kôda po programskim jezicima (2018)

File	files	blank	comment	code
F18_kalk.cloc	141	14476	4686	37531
F18_common.cloc	184	13951	5711	30957
F18_fin.cloc	98	9683	3001	25253
F18_ld.cloc	64	8053	2344	20606
F18_fakt.cloc	61	7147	2305	17117
F18_pos.cloc	60	5249	1893	13241
F18_core.cloc	46	2855	679	6249
F18_epdv.cloc	38	2852	1002	5761
F18_mat.cloc	21	1947	620	5454
F18_os.cloc	21	1706	610	4419
F18_fiskalizacija.cloc	9	2031	1002	3915
F18_SQL.cloc	6	1819	1542	2990
F18_virm.cloc	14	1061	319	2094
F18_core_semafori.cloc	7	780	248	1787
F18_scripts.cloc	37	384	86	743
SUM:	807	73994	26048	178117

Tabela 5: Broj linija programskog koda po programskim modulima (2018)

### 8.3.3 Zaključak

Dobijeni rezultati potvrđuju da su se desile radikalne promjene strukture izvornog kôda. Refaktoring i “čišćenje” nepotrebnog koda (eng. cleanup) učinile su projekat značajno jednostavnijim za održavanje i nadogradnju. Ove aktivnosti su u kontekstu agilnog razvoja softvera realizacija principa 8 koji navode tvorci Manifesta: “*Agilni procesi promoviraju održivi razvoj softvera. Investitori, developeri, korisnici trebaju moći nastaviti učešće u održavanju softvera neograničeno.*”

“F18” je zbog poduzetih aktivnosti postao projekat koji se uz manje troškove može održavati i nadograđivati.

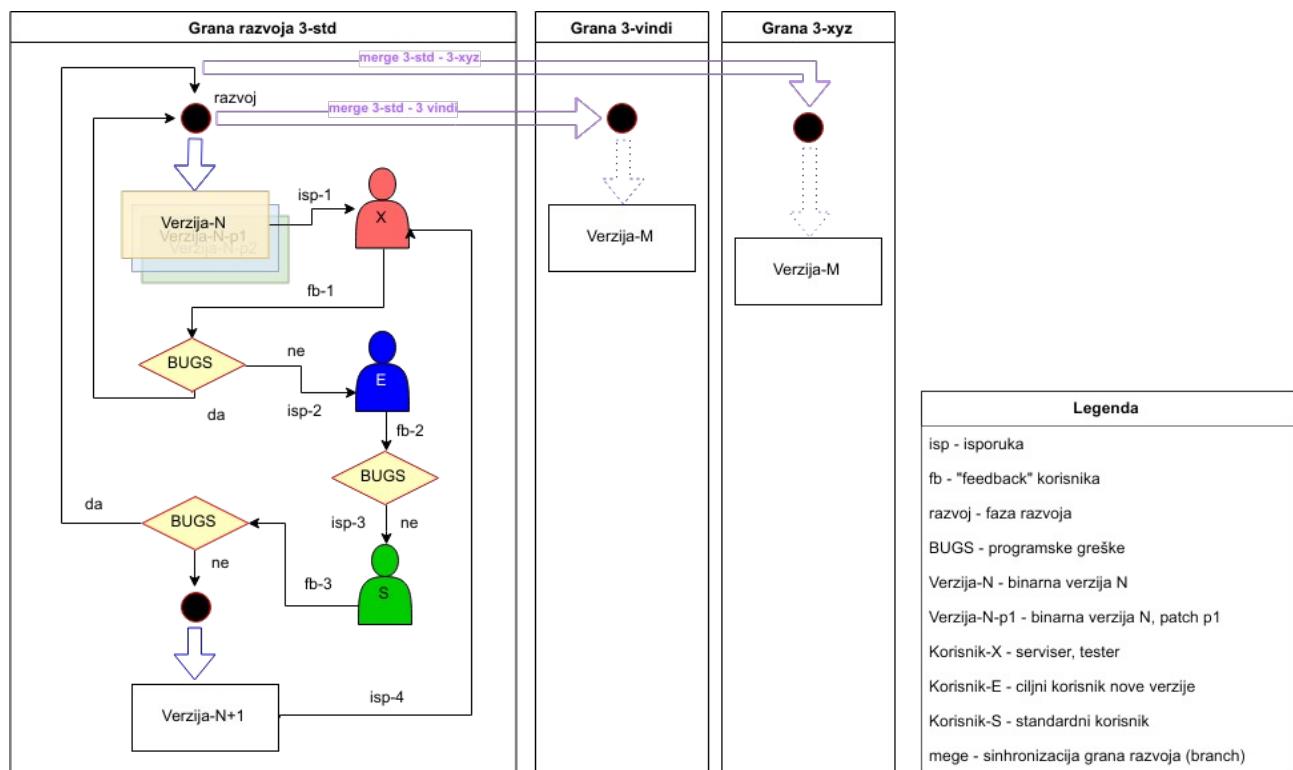
Što se tiče pojedinačnih metoda agilnog razvoja, podsjetimo se da je centralna ideja “*Lean software development*”-a upravo eliminacija “otpada”. Stoga se može reći da su primjenjene “lean” tehnike.

## 8.4 “F18 update” - provjera i instalacija novih verzija

U posljednjoj liniji zajedničkih parametara, navedeno je da klijent kontroliše dostupnost novih verzija, te da koristi granu “3-std” (eng. git-branch), standardnu verziju (mogući izbori su S-standardna verzija, E-edge, X-eksperimentalna verzija). Prilikom prvog pokretanja aplikacija vrši se provjera stanja na “github” serveru:

Download VERSION  
[https://raw.githubusercontent.com/knowhow/F18\\_knowhow/3-std/VERSION](https://raw.githubusercontent.com/knowhow/F18_knowhow/3-std/VERSION)

Ovaj koncept automatske instalacije novih verzija se pokazao veoma bitnim. On je omogućio da samo ciljana grupa korisnika dobija najnovije verzije (E), dok većina dobija verzije sa izvjesnim zakašnjenjem (S). Period “Razvoja -> isporuka nove verzije” je u većini slučajeva značajno smanjen i pojednostavljen. Većina korisnika(S) dobija stabilne verzije, s obzirom da se glavni trijaž grešaka dešava unutar male grupe korisnika (X, E). Dijagram 7. grafički prikazuje tok operacija isporuke:



Dijagram 7: F18 iteracije - isporuka novih verzija korisnicima, tok operacija  
(eng. deployment workflow)

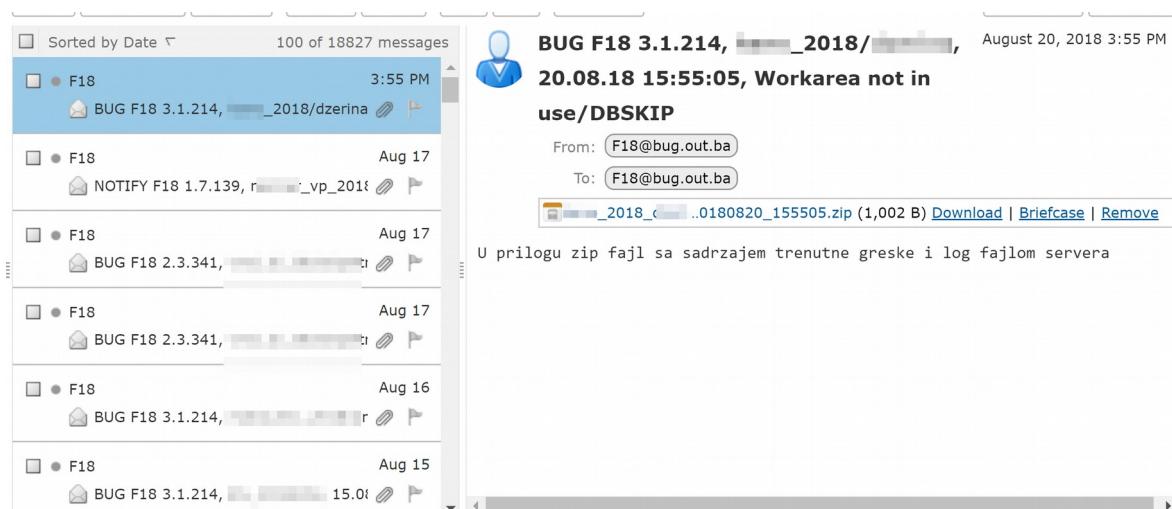
### **8.4.1 Zaključak**

Ova opcija je direktno pomogla agresivnijoj i manje “bolnoj” isporuci novih verzija (sa aspekta korisnika, ali i razvojnog tima). Ona je omogućila da se nove opcije isporučuju svakodnevno, nerijetko više puta na dan. Prije realizacije ove opcije, takva frekvencija izdavanja novih verzija jednostavno nije bila moguća. U poglavlju “Organizacija XP tima” je naglašena **iterativnost razvoja** kao temeljna praksa kojom se postiže agilnost razvoja. Svaka od agilnih metoda ukazuje na temeljnu važnost iterativnog pristupa. To je i očekivano, uvezvi u obzir da je princip 1. tvoraca Manifesta: “*Naš najveći prioritet je zadovoljiti klijenta kroz rane i kontinuirane isporuke funkcionalnog softvera*”.

Pored toga, dijagram toka (Dijagram 7) ukazuje na intenzivne i kratke “feedback” petlje između razvojnog tima i korisnika.

## 8.5 F18 automatski izvještaji o greškama (eng. bug reports)

Na početku 2014 je dizajnirana i realizovana opcija automatske prijave grešaka<sup>39</sup>. Ova opcija na “bring.out” email server šalje izvještaje greškama. U periodu 2014-2018 na server je stiglo skoro **19000 bug reporta**.



Slika 33: Lista bug reporta u email klijentu

Svaki bug report sastoji se od dva priloga (Slika 33):

- Isječak serverskog log-a
- Lista poziva koji su uzrokovali grešku (eng. “call stack”) i stanje bitnih parametara u toku greška

Sadržaj jednog email-a koji prijavljuje grešku pristiglu od korisnika glasi:

**BUG F18 3.1.214, firmatest\_2018/korisnik4, 16.08.18 14:12:38, Zero divisor//**

U naslovu greške nalazi se verzija (3.1.214), baza podataka i poslovna godina (firmatest, 2018), ime korisnika kod koga se desila greška (korisnik4), datum i vrijeme, te poruka o grešci.

39 [https://github.com/knowhow/F18\\_knowhow/blob/3-std/common/bug\\_report.prg](https://github.com/knowhow/F18_knowhow/blob/3-std/common/bug_report.prg)

## 8.5.1 Serverski log

Izgled serverskog log-a prikazan je na Listingu 20:

```
PREGLED LOG-a
-----
Datum / vrijeme   Korisnik   operacija
-----
2018-08-16 12:12:34 korisnik4   GLOBALERRORHANDLER(181) : BUG REPORT: Verzija programa: 3.1.214/5.10.0
                                KALK_STAMPA_DOK_RN / 141 // 3 KALK_STAMPA_DOKUMENTA / 198 ....
2018-08-16 12:06:09 korisnik4   KALK_AZUR_SQL(713) : F18_DOK_OPER: ažuriranje kalk dokumenta: 10-16-G0087/ZS
2018-08-16 12:06:07 korisnik4   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-96-00087/ZS
2018-08-16 12:06:01 korisnik4   KALK_AZUR_SQL(713) : F18_DOK_OPER: ažuriranje kalk dokumenta: 10-96-00087/ZS
2018-08-16 12:04:21 korisnik4   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-UF-00000188
2018-08-16 12:04:09 korisnik4   FIN_NALOG_BRISI_IZ_KUMULATIVA(165) : F18_DOK_OPER: POVRET_FIN: 10-UF-00000188
...
2018-08-16 11:46:41 korisnik4   KALK_AZUR_SQL(713) : F18_DOK_OPER: ažuriranje kalk dokumenta: 10-10-00139/ZS
2018-08-16 11:46:20 korisnik4   KALK_AZUR_SQL(713) : F18_DOK_OPER: ažuriranje kalk dokumenta: 10-10-00138/ZS
2018-08-16 11:39:50 korisnik3   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-66-B0000208
2018-08-16 11:38:10 korisnik3   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-66-B0000209
2018-08-16 11:36:05 korisnik3   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-66-B0000210
...
2018-08-16 10:49:58 korisnik1   KALK_POVRAT_DOKUMENTA(125) : F18_DOK_OPER: KALK DOK_POV: 10-81-00256/BP
2018-08-16 10:41:08 korisnik4   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-IF-00000246
2018-08-16 10:36:55 korisnik4   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-61-ML000152
2018-08-16 10:23:22 korisnik4   FAKT_BRISANJE_PRIPREME(650) : F18_DOK_OPER: fakt, brisanje dokumenta iz pripreme: 10-20-00009
2018-08-16 10:22:29 korisnik4   POVRET_FAKT_DOKUMENTA(153) :
                                F18_DOK_OPER: fakt povrat dokumenta u pripremu: 10-20-00009
...
2018-08-16 10:20:20 korisnik4   BROWSE_SNIMI_PROMJENE_SIFARNIKA(979) :
                                F18_DOK_OPER: dodavanje/ispravka zapisa u sifarnik roba
...
2018-08-16 10:17:04 korisnik4   BROWSE_SNIMI_PROMJENE_SIFARNIKA(1002) : F18_DOK_OPER: NOVI TBL: roba NEW: ID=5010#MATCH_CODE=#SIFRADOB=#NAZ=Test Artikal XYZ#JMJ=KG#IDTARIFA=PDV0#NC=0.00#VPC=11.73#MPC=11.- ...
2018-08-16 10:17:04 korisnik4   BROWSE_SNIMI_PROMJENE_SIFARNIKA(979) : F18_DOK_OPER: dodavanje/ispravka zapisa u sifarnik roba
2018-08-16 10:07:03 korisnik4   BROWSE_SNIMI_PROMJENE_SIFARNIKA(1002) : F18_DOK_OPER: NOVI TBL: partn NEW: ID=P-BAND#MATCH_CODE=#NAZ="TEST PREDUZECE" D00 SARAJEVO#NAZ2=#PTT=71000#MJESTO=SARAJEVO#ADRESA=SARAJEV-O STARI GRAD#ZIROR=#REJON=#TELEFON=099/110-3933#DZIROR=#FAX=#MOBTEL=#IDOPS=#_KUP=#_DOB=#_BANKA=#_RA-DNIK=#IDREFER=#
...
2018-08-16 06:43:18 korisnik1   EDIT_FIN_PRIPR_KEY_HANDLER(386) :
                                F18_DOK_OPER: fin, brisanje stavke u pripremi: 10-61-BP000160      stavka br: 14
2018-08-16 06:36:20 korisnik1   FIN_AZURIRANJE_NALOGA(110) : F18_DOK_OPER: azuriranje fin naloga: 10-61-T0000159
2018-08-16 06:31:33 korisnik2   KALK_AZUR_SQL(713) : F18_DOK_OPER: ažuriranje kalk dokumenta: 10-42-00221/5
```

Listing 20: Serverski log

## 8.5.2 “Call stack” i stanje sistema u trenutku greške

Ovaj izvještaj sadrži (Listinzi 21, 22):

- “call stack”-a
- sadržaj tabele koja je bila otvorena u toku greške (KALK\_PRIPR)
- niz podataka o serveru (verzija, relevantne IP adrese, itd.)

```
F18 bug report (v6.0) : 16.08.18 14:12:35
=====
Verzija programa: 3.1.214/5.10.0 27.06.2018

SubSystem/severity : BASE      2
GenCod/SubCode/OsCode :      5    1340      0
Opis : Zero divisor
ImeFajla :
Operacija : /
Argumenti : _?_
canRetry/canDefault : .f. .f.

BUG REPORT: Verzija programa: 3.1.214/5.10.0 27.06.2018 ;
SubSystem/severity : BASE      2 ;
GenCod/SubCode/OsCode :      5    1340      0 ;
Opis: Zero divisor ; ImeFajla : ;
Operacija: / ; Argumenti : _?_ ;
canRetry/canDefault : .f. .f. ;
CALL STACK:
1 (b)F18_ERROR_BLOCK / 52
2 KALK_STAMPA_DOK_RN / 141
3 KALK_STAMPA_DOKUMENTA / 198
4 KALK_PRIPR_KEY_HANDLER / 235
5 (b)KALK_PRIPR_OBRADA / 141
6 MY_BROWSE_F18_KOMANDE_WITH_MY_KEY_HANDLER / 654
7 MY_BROWSE / 337
8 KALK_PRIPR_OBRADA / 141
9 (b)TKALKMOD_PROGRAMSKI_MODUL_OSNOVNI_MENI / 57
10 F18_MENU / 58

11 TKALKMOD:PROGRAMSKI_MODUL_OSNOVNI_MENI / 92
12 TKALKMOD:MMENU / 44
13 TKALKMOD:RUN / 126
14 MAINKALK / 27
15 (b)SET_PROGRAM_MODULE_MENU / 316
16 F18_PROGRAMSKI_MODULI_MENI / 274
17 F18_LOGIN_LOOP / 93
18 MAIN / 45
-----
/---- SERVER connection info: ----- /
host/database/port/schema : 192.168.59.10      /
firmatest_2018 / 5432 / public
user : korisnik4

F18 client required server db >= : 0.0.25 / 25
Actual knowhow ERP server db version : 0.0.27 / 27
/----- BEGIN PostgreSQL vars -----/
server_version : 9.5.1
TimeZone : UTC
/----- END PostgreSQL vars -----/
client_addr : 192.168.59.114
client_port : 53716
server_addr : 192.168.59.10
server_port : 5432
user : korisnik4

/----- END PostgreSQL sys info -----/
```

Listing 21: "Call stack" i parametri sistema (prvi dio)

Record content:

1 IDFIRMA	C	2	0 10		27 MPCSAPP	N	18	8	0.00000000
2 IDRBA	C	10	0 9960		28 IDTARIFA	C	6	0	PDV17
3 IDKONTO	C	7	0 1200		29 MKONTO	C	7	0	1200
4 IDKONTO2	C	7	0 1100		30 PKONTO	C	7	0	
5 IDZADUZ	C	6	0		31 MU_I	C	1	0	1
6 IDZADUZ2	C	6	0 087/18		32 PU_I	C	1	0	
7 IDVD	C	2	0 RN		33 ERROR	C	1	0	0
8 BRDOK	C	8	0 ,0087/PP		34 KOLICINA	N	18	8	3.52000000
9 DATDOK	D	8	0 11.08.18		35 GKOLICINA	N	18	8	0.00000000
10 BRFAKTP	C	10	0 087/18		36 GKOLICIN2	N	18	8	0.00000000
11 DATFAKTP	D	8	0 11.08.18		37 FCJ	N	18	8	0.00000000
12 IDPARTNER	C	6	0		38 FCJ2	N	18	8	0.00000000
13 RBR	C	3	0 6		39 FCJ3	N	18	8	0.00000000
14 PODBR	C	2	0		40 RABAT	N	18	8	0.00000000
...					...				
21 TRABAT	C	1	0		46 ZAVTR	N	18	8	0.00000000
22 TMARZA	C	1	0 A		47 MARZA	N	18	8	-9999.00000000
23 TMARZA2	C	1	0		48 MARZA2	N	18	8	0.00000000
24 NC	N	18	8 9999.00000000		49 RABATV	N	18	8	0.00000000
25 MPC	N	18	8 0.00000000		50 VPCSAP	N	18	8	0.00000000
26 VPC	N	18	8 0.00000000						

18 MAIN / 45  
== END OF BUG REPORT ==

Listing 22: "Call stack" i parametri sistema (drugi dio)

### 8.5.3 Svrha i značaj automatskih "bug report"-a

Motiv za izradu automatskih bug-reporta je bila činjenica da je do tada serviseru bilo potrebno 30-45 minuta da prikupi podatke o grešci. Povećani razvoj redovno rezultira povećanjem programskih grešaka. U najvećem broju slučajeva, greška bi bila otklonjena u kratkom roku nakon što se dobiju podaci o grešci. Međutim, ručno prikupljanje tih podataka bi se često znalo desiti tek nekoliko dana nakon prijave problema. To je stvaralo dva glavna problema:

- serviser je izložen velikom pritisku
- korisnici nezadovoljni sporim rješavanjem problema

Oba problema su eliminisana automatskim bug reportima. Jednostavan ekonomski račun efekata ove opcije se izračunava na sljedeći način:

- Sa pretpostavkom da su pristigli bug report-i imali koeficijent ponavljanja 4, te da prosječno ručno prikupljanje traje 1/2 h, direktna ušteda je  $19000/4*0,5 = 2375$  serviser/h.
- Kada se taj iznos podijeli na 4 godine, u svakom mjesecu je smanjen angažman servisera za 50 h.

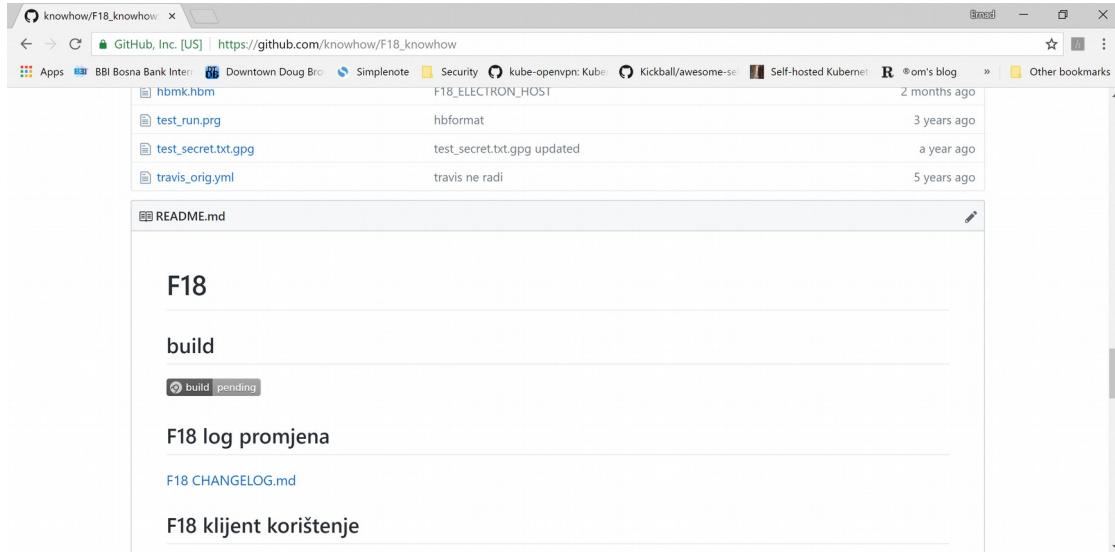
Može se zaključiti da su automatski reporti proizveli korist koja odgovara angažmanu jednog dodatnog servisera!

### 8.5.4 Zaključak

Veliki dio zaključaka koji se odnose na ranije navedenu opciju "F18 update" su identični u slučaju ove opcije. Ove dvije opcije su svakako komplementarne i usko povezane. Tome se može dodati (terminologijom "Lean software development"-a) da je uz pomoć ove opcije eliminisan "otpad" nepotrebnih operacija servisera.

## 8.6 F18 automatska integracija (CI)

Automatska integracija (eng. Continous integration - CI) koja se koristi u "F18" obezbjeđuje da se nakon publikovanja nove verzije projekta na "github", aktivira automatsko kreiranje Linux i Windows verzije. Kao CI platforma koristi se "Appveyor" servis<sup>40</sup>:



Slika 34: Prikaz stanja "appveyor CI" na "github" stranici projekta ("build pending" status - kreiranje nove verzije je započelo)

Kada se na git repozitorij na "github"-u publikuje nova verzija (sa git "tag"-om i predefinisanom porukom "BUILD\_RELEASE XYZ") započinje build proces a "appveyor" serverima (Slika 34). Ako je "build" proces uspješan, artifakt se publikuje na "bintray" (Slika 35):

```
1430 ++ [[ $hx8c != *i* ]]
1431 ++ return
1432 + '[' -n /mingw32/share/aclocal:/usr/share/aclocal ']'
1433 + export ACLOCAL_PATH
1434 + export PATH MANPATH INFOPATH PKG_CONFIG_PATH USER TMP TEMP PRINTER HOSTNAME PS1 SHELL tmp temp ORIGINAL_TMP ORIGINAL_TEMP
ORIGINAL_PATH
1435 + unset PATH_SEPARATOR
1436 + '[' $false = true ']'
1437 + unset MAYBE_FIRST_START
1438 + '[' -f /home/appveyor/.bashrc ']'
1439 + source /home/appveyor/.bashrc
1440 ++ [[ $hx8c != *i* ]]
1441 ++ return
1442 + zip F18_windows_x86_3.1.215.zip F18.exe
1443 adding: F18.exe (deflated 65%)
1444 Start-Service postgresql-x64-9.6
1445 SET PGUSER=postgres
1446 SET PGPASSWORD=Password12!
1447 PATH=C:\Program Files\PostgreSQL\9.6\bin;%PATH%
1448 createdb F18_test
1449 cd %APPVEYOR_BUILD_FOLDER%
1450 F18.exe -h localhost -u postgres -p Password12! -d F18_test --show-postgresql-version
PostgreSQL 9.6.9, compiled by Visual C++ build 1800, 64-bit
1451 collecting artifacts...
1452 Found artifact 'F18_windows_x86_3.1.215.zip' matching 'F18_windows_x86_3.1.215.zip' path
1453 Uploading artifacts...
1454 [1/1] F18_windows_x86_3.1.215.zip (3,570,801 bytes)...100%
1455 "BinTray" deployment has been skipped as environment variable has not matched ("BUILD_ARTIFACT" is "windows_x86", should be "linux_x86")
1456 Deploying using BinTray provider
1457 Uploading "F18_windows_x86_3.1.215.zip" to BinTray as hernad/F18/F18-windows-x86/3.1.215/F18_windows_x86_3.1.215.zip...OK
1458 if [ $BUILD_TYPE == "Release" ] echo "SUCCESS"
1459 Build success
```

Slika 35: "Appveyor" detalji "build" procesa

40 <https://www.appveyor.com>; Appveyor je besplatan za OSS projekte.

“Bintray”<sup>41</sup> je servis na koji se publikuju binarne verzije (Slika 36):

The screenshot shows a web browser window with the URL <https://bintray.com/hernad/F18>. The page displays four package entries:

- F18-linux-x86**: Version 3.1.214 (Jun 27, 2018) by Ernad Husremović. Rating: ★★★★★.
- F18-windows-x86**: Version 3.1.215 (Aug 13, 2018) by Ernad Husremović. Rating: ★★★★★.
- postgresql-windows-x86-dlls**: Version 3.1.215 (Aug 13, 2018) by Ernad Husremović. Rating: ★★★★★.
- template**: Version 3.1.215 (Aug 13, 2018) by Ernad Husremović. Rating: ★★★★★.

A search bar at the top left contains "Package Name". To the right of the search bar are sorting options: "Sorted By Name" and a green checkmark icon. A blue wrench icon is located in the top right corner of the main content area. A vertical sidebar on the right is titled "Feedback".

Slika 36: web stranica “bintray” servisa, windows verzija (3.1.215) publikovana.

Identičnim “build” procesom se kreiraju i linux verzije. Po uspješnom okončanju čitavog procesa, “appveyor” kontrolna ploča daje status “zeleno” (Slika 37):

The screenshot shows a web browser window with the URL <https://ci.appveyor.com/project/hernad/f18-knowhow>. The page displays the “LATEST BUILD” section for the project “F18\_knowhow”. The build status is “3.1.215” with a green progress bar indicating it was completed 12 minutes ago by Ernad Husremovic. The build duration was 3 minutes and 22 seconds. Below the build details, there is a table showing environment variables and their values:

JOB NAME	TESTS	DURATION
Environment: APPVEYOR_BUILD_WORKER_IMAGE=Visual Studio 2015, MSYS2_ARCH=i686, MSYSTEM=MINGW32, BUILD_A...	33	4 min 5 sec
Environment: APPVEYOR_BUILD_WORKER_IMAGE=Ubuntu, BUILD_ARTIFACT=linux_x86, appveyor_repo_tag=true		2 min 7 sec

Slika 37: “Appveyor” kontrolna ploča na kraju uspješnog “build” procesa

41 Slično “appveyor”-u, “bintray” je besplatan za “open-source” projekte.

## 8.6.1 Zaključak

Nekoliko minuta nakon publikovanja izvornog kôda nove verzije su dostupne korisnicima. Automatizirani proces kreiranja binarnih verzija obezbjeđuje redovno integrise tehnike automatskog testiranja prije publikovanja novih verzija. Iako "F18" nije pokriven detaljnim funkcionalnim testovima (nisu prakticirane TDD tehnike), appveyor izvršna skripta<sup>42</sup> provjerava da li kreirana verzija uspješno komunicira sa database serverom (Tabela 6):

test_script:	
	- cmd: cd %APPVEYOR_BUILD_FOLDER%
Windows =>	- cmd: F18.exe -h localhost -u postgres -p Password12! -d F18_test --show-postgresql-version
Linux =>	- sh: LD_LIBRARY_PATH=. xvfb-run --server-args="-screen 0 1024x768x24" ./F18 -h localhost -u postgres -p Password12! -d F18_test --show-postgresql-version

Tabela 6: test komande unutar "appveyor" izvršne skripte

Bez obzira na jednostavnost, ovaj test garantuje da se korisniku neće isporučiti verzija koja ne može komunicirati sa bazom podataka, što je od suštinske važnosti. Automatsko kreiranje novih binarnih verzija (CI - eng. continuous integration) je tehnika koja je u svim agilnim metodama visoko preporučena, a u XP obavezna. U "*Lean software development*" metodi CI je praksa koja razvojni tim rješava nepotrebnih operacija manuelne isporuke nove verzije.

U kontekstu projekta "F18", očigledna je povezanost ove opcije sa "F18 update" opcijom. Bez ovih tehnika **iterativni razvoj** bio bi praktično nemoguć.

42 [https://github.com/knowhow/F18\\_knowhow/blob/3-std/appveyor.yml](https://github.com/knowhow/F18_knowhow/blob/3-std/appveyor.yml)

## **9 Iz osobne prakse**

### **9.1 Software koji se ne koristi je otpad**

Softverska industrija svakodnevno izbacuje nove alate i tehnologije. Upoznavanje sa novim tehnologijama je potrebno i poželjno. Međutim, ako se nove tehnologije bez opravdanih razloga integriraju u proizvode dobijamo otpad.

*Primjer:*

Developer procjeni da mu “Python runtime” može biti od velike pomoći u budućnosti, te stoga daje uputu serviserima da instaliraju “Python” na svaku radnu stanicu kod korisnika.

Iako je prednost ove aktivnosti za developera važna, gubici koje nije uzeo u obzir su sljedeći:

- Serviser gubi dodatno vrijeme na instalaciju softvera koji korisniku nije potreban
- Taj softver nepotrebno zauzima prostor
- Svaki komad softvera je predmet budućeg održavanja i moguća sigurnosna prijetnja

Ovo je tipični primjer otpada u Lean tehnologiji.

## 10 Zaključak

Agilni manifest je objavljen prije 17 godina. Danas se može reći da su agilne metode postale prije standard negoli novitet u organizaciji softverskih projekata i razvojnih timova. Agilni razvoj prepoznaje specifičnosti razvoja softvera u odnosu na druge inžinjerske grane. Dinamičnost industrije informacionih tehnologija određuje da učenje i usvajanje novih znanja bude ne samo put ka izvrsnosti, nego i uslov opstanka. Agilni razvoj je rezultat traganja za konceptima te činjenice uvažavajući radnu psihologiju. One su stoga koliko tehničke toliko i emocionalne - bave se programerskim tehnikama, ali i motivacijom, ugodnom radnom atmosferom, osjećajem postignuća.

U praktičnom dijelu rada prikazan je razvojni ciklus projekta "F18". Navedeni su motivi, tehnički i poslovni ciljevi, problemi, te postignuti rezultati u razmatranom periodu. "F18" je određen sljedećim faktorima:

- projekat ima 25-godišnju istoriju, što ga u informatici čini starim
- ciljna industrija su finansije, knjigovodstvo i podrška poslovanju
- ciljna klijentela je lokalna, tržište malo (mala kupovna moć)
- razvojni tim je mali (dva čovjeka)

Ovi faktori određuju širi kontekst projekta. Na osnovu njih se mogu procijeniti mogućnosti daljeg razvoja, toka i održivosti projekta. Bilo bi pogrešno ocijeniti razmatrani razvojni period kao period u kome se prešlo (ili nije prešlo) na agilni razvoj softvera. Projekat je u ovom periodu prošao kroz egzistencijalnu krizu. U tom periodu se nije tražio odgovor na optimalan način razvoja, nego odgovor na pitanje "biti ili ne biti". Iako su ovo pitanja na koje bi glavnu riječ trebali voditi ekonomisti a ne informatičari, ovakve situacije i dileme su od temeljne važnosti za projekat i razvojni tim. Ako softverski projekat izgubi elemente održivosti, on propada bez obzira na kvalitet programske kôde i mogućnosti razvojnog tima. Princip 8. autora agilnog Manifesta to naglašava: "*Agilni procesi promoviraju održivi razvoj softvera. Investitori, developeri, korisnici trebaju moći nastaviti učešće u održavanju softvera neograničeno*".

Mnogobrojni nedostaci i problemi projekta "F18" nisu otklonjeni i ostati će ograničavajući faktor razvoja. Svaki softverski proizvod je kombinacija mogućeg, a ne želenog. On je dobar i funkcioniše u onoj mjeri u kojoj ga takvim **smatraju klijenti koji ga koriste**. Za razvojni tim, odnosno preduzeće koje proizvodi softver, mjera uspjeha su poslovni rezultat, te novi projekti koji nastaju na bazi postojećih. Uvezši u obzir ove parametre, razvojni period 2014-2018. se može ocijeniti "F18" uspješnim.

Za svaku od prezentovanih aktivnosti na "F18" su napravljeni posebni podzaključci u kontekstu primjene agilnih koncepata. Unutar tih podzaključaka najviše referiranja je bilo na "*Lean software development*" metode. Dugogodišnja istorija i stanje projekta na početku razvojnog ciklusa su odredili prioritete, a to su jezikom "lean" bile eliminacija **otpada** i podizanje **kvaliteta** postojeće programske baze.

Najvažniji nedostatak, koji je ostao prisutan i nakon završetka razvojnog ciklusa, je nedostatak automatskih testova programske baze. To je onemogućilo primjenu TDD praksi<sup>43</sup> i veći kvalitet CI procesa. Taj nedostatak je ublažen opcijama “F18 update” i “F18 bug reports”<sup>44</sup>, ali one ipak ne mogu zamijeniti testove. Kreiranje seta testova na “F18” kodu je razmatrano ali je na kraju ipak odbačeno radi malog kapaciteta razvojnog tima naspram velike programske baze, arhitekture aplikacije i ograničenja alata za razvoj<sup>45</sup>.

Pozitivni rezultati predhodnog su stvorili pretpostavke da se otvori novi razvojni ciklus. Ranija iskustva i novopostavljeni ciljevi svakako će biti ulazni parametri novog razvojnog ciklusa. Očekivanja su da će rezultati istraživanja iz ovog rada biti od konkretne pomoći razvojnom timu “bring.out”, ali i drugim razvojnim timovima koji rade na sličnim projektima.

---

43 Ref. paragraf: 2.5

44 Ref. parografi: 8.4, 8.5

45 “harbour” programski jezik nema ugrađenu podršku za formiranje testova kao što to imaju drugi programski jezici

## Literatura

AART: James Shore & Shane Waden, The Art of Agile development, 2008

APRAT: Venkat Subramaniam & Andy Hunt, Practices of an Agile Developer, 2006

ASAM: Jonathan Rasmusson, The Agile Samurai, 2014

PROGIT: Scott Chacon & Ben Straub, Pro Git, 2nd Edition, 2014

LEARNAG: Andrew Stellman & Jennifer Greene, Learning Agile, 2015

SEDAWK: Dale Dougherty and Arnold Robbins, sed & awk, Second Edition, 1997

## Korišteni softver:

- Windows 10 Home
  - msys2 <https://www.msys2.org>
- Fedora Workstation 28
  - Gnome terminal 3.28
- LibreOffice calc, writer 6.0
- Gimp 2.8
- Pygments 2.2
- Git 2.17
- cloc
  - <https://github.com/AlDanial/cloc>
- draw.io desktop 9.1
- Atom 1.30
- harbour 3.4.6hernad (0ed48dfa0b) (2018-04-05 12:24)
  - <https://github.com/hernad/harbour-core>