Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16, Bratislava 4

Deep Search
Metodiky vývoja

Vedúci tímu: Ing. Nadežda Andrejčíkova, PhD.
Členovia tímu: Bc. Peter Berta, Bc. Matej Adamov, Bc. Michal Krempaský, Bc. Oliver Macko, Bc. Bronislava Pečíková
Akademický rok : 2017/2018

Obsah

1. Metodiky verziovania

Pre správu verziovania projektu používame nástroj *Git*, konkrétne aplikáciu *GitLab*. Vytvorili sme si 3 hlavné repozitáre:

- neverest/deepsearch – samotná aplikácia projektu
- neverest/neverest.gitlab.io – webová stránka tímu
- neverest/docs – dokumentácia k projektu

Existuje množstvo pracovných postupov (*workflow*) ako pracovať s gitom. My sme si zvolili štandardne zaužívaný postup s názvom *Gitflow*[1].

Základom projektu je hlavná (*master*) vetva (*branch*). V tejto vetve držíme len stabilnú a funkčnú verziu aplikácie. Nad touto vetvou je vytvorená vetva *develop*, v ktorej sa nachádza pracovná verzia aplikácie. Táto verzia musí byť skompilovateľná a musia v nej prejsť všetky testy, nemusí však nutne obsahovať správnu alebo funkčnú implementáciu danej funkcionality. Na konci šprintu sa najnovšia funkčná verzia *developu* pripojí (*merge*) k *masteru*. Nad *developom* sa potom vytvárajú vetvy na implementáciu konkrétnej funkcionality. Názov vetvy pozostáva z prefixu, id úlohy v *ScrumDesku* a názvu, prípadne stručného popisu implementovanej funkcionality.

Prefix sa určuje podľa typu zmeny, ktorú ideme implementovať. Určili sme si 3 základné prefixy:

- *feature* – implementácia novej funkcionality
- *bugfix* – oprava chyby v už implementovanej funkcionalite
- *hotfix* – špeciálny typ prefixu, ktorému by sme sa čo najradšej vyhli, jedná sa o opravu chyby v *master* vetve, ktorá znemožňuje funkčnosť celej aplikácie alebo jednej z hlavných funkcionalít. Takáto vetva sa vytvorí priamo nad *mastrom* a pripojí sa do *mastra* aj počas behu šprintu.

Príklad názvu *branche*: feature/#1234/tokenizacia-textu

Po dokončení práce na svojej vetve si developer vytvorí *pull request* (nazývané aj *merge request*) do *develop* vetvy. *GitLab* následne nad týmto *pull requestom* vykoná nastavené testy (CI). Ak testy prebehnú úspešne, *pull request* je pripravený na schválenie (*code review*) ostatnými členmi tímu. Ak sa tím zhodne, že je implementovaná funkcionalita v poriadku, *pull request* môže byť pripojený do *developu*. Dohodli sme sa, že *pull request* musia schváliť minimálne ďalší dvaja členovia tímu na to aby mohol byť *mergnutý* do *developu*. Výnimku tvorí *hotfix*, ktorý musia schváliť až traja keďže ide o zásah priamo do *master* vetvy.

[1] https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

2. Metodika testovania a písania testov

Koncept testovania v našom projekte delíme v rámci automatizovaných testov na jednotkové(*unit*), integračné a biznis(funkčné) testy. Okrem iného testujeme aj manuálne.

2.1. Jednotkové testy

Jednotkové testy sa snažíme písať pre každú metódu samostatnú test metódu, kde sa snažíme test čo najviac izolovať od zvyšku systému. Najideálnejšie sa testujú metódy, ktoré vrátia objekt, ktorý vieme porovnať so želaným objektom.

Metodika jednotkových testov je v tomto jasne definovaná, aby vývojár napísal jednotkový test potvrdzujúci jeho vlastné pochopenie špecifikácie dodávanej funkcionality. To zahŕňa implementáciu minimálne jedného pozitívneho scenára, minimálne jedného negatívneho scenára uvažujúci vstupy, ktoré nie sú v súlade so špecifikáciou. Tieto testovacie metódy implementujeme v balíku “tests”. Metódy, triedy a scripty organizujeme do balíkov podľa štruktúry zdrojového kódu v zmysle, ak sú triedy organizované v nejakej hierarchii balíkov, tak názvy balíkov sú odzrkadlené v hlavnom balíku “tests”. Pred každý script napíšeme prefix test_script.py.

Projekt:

- process_text.py
- pipeline.py

Tests:

- test_process_text.py
- test_pipeline.py

Tieto testy voláme cez knižnicu “pytest”.

2.2. Integračné testy

Integračné testy sú primárne implementované pre moduly, ktoré sa navzájom po sebe volajú, aby sme zabezpečili správne integrovanie. Tieto testy sú implementované na základe simulácie externých volaní daného modulu, ale aj prepojenia funkcionalít v rámci modulu. Implementácia týchto testov odzrkadľuje potrebu testovania kritických modulov a schopnosť prevádzky biznis logiky systému. Sústredenie týchto testov spočíva v odhalení čo najviac kritických chýb v menšom počte testov, z dôvodu údržby. Implementujeme ich podľa podobnej konvencie ako jednotkové testy s tým rozdielom, že balík je nazvaný “int_tests” a scripty nemusia odzrkadľovať štruktúru projektu. Pre zachovanie dobrého organizovania týchto testov je vhodné pomenovať tento test v kombinácií skriptov, poprípade vymyslieť stručný názov.

Projekt:

- process.py
- pipeline.py
- store.py

Int_tests:

- process_store.py

2.3. Biznis testy

Vhodné biznis testy vznikajú v prípade, že máme jasný prípad použitia s presne zadefinovanými krokmi. Tento test slúži na otestovanie základnej prevádzky schopnosti systému. Ak nezbehne tento test, v tom prípade je možné usúdiť, že produkt nespĺňa základnú požiadavku na použitie. Tento test sa vykonáva po zbehnutí jednotkových testov a integračných testov.

2.4. Manuálne testy

Metodika manuálneho testovania popisuje len v krátkosti účelovosť a potrebu tohto testovania.

Hlavnou pointou manuálneho testovania je objavenie nelogických krokov, ktoré môžu vyplynúť z nedostatočného pochopenia požiadaviek. Manuálne testovanie v sebe zahŕňa aj vyhľadávanie kritických stavov, do ktorých sa môže program dostať. Rovnako sem patrí aj vytváranie negatívnych vstupov, ktoré môžu poškodiť systém. Táto metóda testovania má pokryť aj také prípady, v ktorých by bolo veľmi náročné implementovať automatizované testy.

3. Metodika kontroly kvality

Kvalita v našom projekte je reprezentovaná viacerými praktikami. Väčšina týchto praktík sa zameriava hlavne na písanie dobrého zdrojového kódu.

Dodržiavanie kvality kódu sa dosahuje viacerými prístupmi. Najdôležitejší prístup je revidovanie kódu. Zvyšné prístupy definujú konvencie písania kódu, prístupy ku refaktorovaniu a monitorovanie pokrytia testami.

3.1. Konvencie

Kód vyvíjaný v našom projekte musí spĺňať normu písania zdrojového kódu *PEP8*. Táto norma definuje jasne konvencie písania kódu a nazývania skriptov, tried, metód a premenných. Normu *PEP8* vieme nastaviť do takmer každého vývojového prostredia vrátane prostredia *Pycharm*.

Čitateľnosť kódu sa zabezpečuje vhodným názvoslovím a komentármi. Názvoslovie by malo zodpovedať kľúčovým slovám funkcionality. Pre príklad, ak ukladám dáta do databázy, tak metóda sa má volať v anglickom jazyku slovom zodpovedajúcim prekladu ako *store_data*. Vývojár sa má snažiť pomenovať názvy tak, aby bolo na prvý pohľad jasné čo daná časť kódu rieši.

3.2. Revidovanie kódu

Revidovanie kódu je po naimplementovaných jednotkových testoch prvá dôležitá kontrola naimplementovania funkcionality. Revidovanie je zabezpečené minimálne jedným členom tímu. Členovia tímu sú informovaní o ukončení implementácie vývojárom, ktorý ich požiada cez informačný kanál *Discord* aby mu revidovali kód. Do kanálu pošle odkaz na *merge request* v *GitLabe*. V tomto linku je možná diskusia, kde môžu ľudia vykonávajúci revíziu písať pripomienky. Revízia je úspešná ak vývojár zapracuje všetky pripomienky.

Minimálne podmienky, ktoré musí vývojár splniť predtým než kontaktuje ostatných členov tímu sú:

- ak je implementovaná nová funkcionalita tak musí obsahovať minimálny počet jednotkových testov podľa metodiky testovania
- ak má riešenie viac *commitov*, musia byť *squash*-nute do jedného *commitu*
- kód spĺňa základné konvencie písania kódu PEP8

Revízne komentáre, ktoré je nutné zapracovať:

- jednotkové testy nespĺňaniu minimum kontroly implementácie v zmysle, že nie sú kvalitatívne postačujúce
- kód je nezrozumiteľne napísaný
- kód neošetruje kritické stavy
- kód nespĺňa implementáciu špecifikácie
- kód obsahuje chyby, na ktoré vývojár neprišiel

Revízne komentáre, ktoré nie je nutné zapracovať a môžu byť predmetom ďalších úloh mimo riešenie implementovaného v *commite*:

- neexistujúce integračné testy
- malý počet *unit testov*
- potenciál pre refaktorovanie
- kód nespĺňa konvencie dohodnuté pre kvalitu kódu
- kód nie je testovateľný

3.3. Refaktorizácia

Refaktorizácia je proces, ktorý vykonávame pri nedostatočne optimalizovaných častiach zdrojového kódu, prípadne ak štruktúra kódu zhoršuje pridávanie novej funkcionality. V takýchto prípadoch tím zhodnotí potrebu refaktorizácie modulov a naplánuje to vo forme úloh pre ďalší šprint, kde jasne definuje akým spôsobom sa má daný modul prerobiť.

4. Metodika identifikácie rizík a ich riešenia

Pri projekte a jeho riadení môžeme uvažovať o situáciách, ktoré môžu ohroziť projekt alebo dodávku funkcionality. Tieto situácie identifikujeme ako riziká a metodika definuje postup pri ich objavovaní a riešení.

4.1. Identifikácia rizík

Pri vytváraní úloh sa snaží tím analyzovať možné výskyty rizík počas šprintu. Každý člen sa snaží identifikovať nejaké riziko a potom ho prednesie zvyšku tímu. Tieto riziká kontroluje manažér rizík a ak je argumentácia pre riziko postačujúca, tak navrhne prediskutovať riziko v tíme. Riziká musí ohodnotiť metrikami, akými sú dopad rizika a pravdepodobnosť jeho výskytu. Riziká, ktoré nastanú, je potrebné dokumentovať v rámci aplikácií manažmentov. Riziká, ktoré nenastali, môže manažér rizík rovnako dokumentovať v aplikácií manažmentov. Zdokumentované riziká môže opakovane predkladať tímu na stretnutiach aby motivoval kolegov k hľadaniu potencionálnych problémov.

4.2. Zmiernenie rizík

Identifikované riziko sa následne prediskutuje v tíme. Ak sa tím zhodne na neuvažovaní rizika, tak sa riziko ignoruje. Pokiaľ sa ale rozhodneme riziko zvážiť a prijať, tak hľadáme možnosti ako zmierniť dopad rizika. Manažér rizika je povinný uviesť v diskusii aspoň jeden spôsob ako zmierniť dopad rizika alebo zamedziť výskytu rizika. Proces končí vtedy, keď je vydaný jasný postup spracovania rizika. Tento proces sa následne musí zdokumentovať v aplikácií manažmentov, aby tím vedel postupovať v budúcnosti pri výskyte podobného rizika, prípadne vylepšiť proces zmiernenie rizík.

4.3. Riešenie neočakávaných rizík

Riziká, ktoré sme pri procese plánovania a vytvárania úloh identifikovali sa snažíme urýchlene vyriešiť. Tento typ rizika nastáva počas šprintu a pri jeho objavení musí manažér okamžite problém riešiť. Najskôr sa zistí dopad rizika na ukončenie šprintu, následne sa zhodnotia a aplikujú možné zmiernenia rizík. O vzniknutom riziku je informovaný zvyšok tímu a zhodnotí sa neskôr v retrospektíve šprintu.

5. Metodika pridávania novej technológie a procesu vývoja

Pre pridávanie a definovanie používania novej technológie alebo spôsobu vývoja, je potrebné definovať proces, v ktorom zvyšok tímu schváli pridanie novej technológie.

Člen tímu, ktorý je presvedčený, že nová technológia je prínosom pre tím a vývoj, musí vypracovať základné demo používania technológie a vypracovať analýzu iných riešení v oblasti problematiky. V dokumente následne uvedie výhody a nevýhody používania.

Následne vedie diskusiu s tímom o nasadení technológie. V nej sa snaží presvedčiť členov tímu o výhodách používania technológie alebo procesu vývoja. K technológií musí členom tímu ukázať návody a dokumentáciu, prípadne je pripravený spraviť základné školenie používania.

Pre nové technológie s ktorými nemá skúsenosť žiaden člen tímu, platí povinnosť diskusie. V nej sa snažia hľadať pozitíva a negatíva nového prístupu a používania novej technológie. Následne sa schváli priestor na analýzu novej technológie, ktorej súčasťou je dokumentovanie ako pri overenej technológii.

6. Metodika komunikácie

Správna komunikácia je jeden z najdôležitejších faktorov ovplyvňujúcich dobré fungovanie tímu. Pre zabezpečenie kvality komunikácie je potrebná definícia postupov a pravidiel jej priebehu. Tejto problematike sa venuje metodika komunikácie, ktorá je určená pre všetkých členov tímu.

6.1. Komunikačný nástroj - *Discord*

Ako hlavný komunikačný nástroj si tím vybral voľne dostupný Discord. Tento nástroj existuje vo forme webovej či mobilnej aplikácie, a taktiež aj klasickej inštalovateľnej počítačovej aplikácie.

Tento nástroj umožňuje priamu komunikáciu medzi jednotlivými členmi tímu, vytváranie skupín či vytváranie komunikačných kanálov. Tieto kanály oddeľujú komunikáciu podľa obsahu, čo pomáha k jej štruktúrovanosti. Takto štruktúrovaná komunikácia je mnohonásobne prehľadnejšia a spätné vyhľadávanie je jednoduchšie.

6.2. Kanály

Samotné kanály sú explicitne pomenované podľa obsahu, ktorému sa venujú. Kanály môže vytvárať iba *scrum-master* a manažér komunikácie. Pri potrebe nového kanálu je preto nutné podať požiadavku primárne na manažéra komunikácie.

Aktuálne kanály:

- #general - všeobecná komunikácia
- #to-do - monitorovanie dôležitých a aktuálnych odovzdaní
- #web - konzultovanie a rozhodovanie o webe.
- #linky-doc - obsahuje odkazy na stránky a dokumenty
- #poznamky-zo-stretnuti - obsahuje poznámky zo stretnutí pre rýchlejší prístup
- #dokumentacia - komunikácia ohľadom dokumentácie
- #pre-nadu - otázky a požiadavky na vedúceho práce.

INFO:

- #info-o-predmete - obsahuje informácie o predmete Tímový projekt
- #info-o-time - obsahuje kontaktné údaje na členov tímu a role v tíme.
- #info-o-discorde - informácie o aktuálnych kanáloch

TECHNOLOGIE

- #technologie-tutorialy - obsahuje linky na návody rôznych technológií.
- #technologie-diskusia - všeobecná komunikácia ohľadom technológií.

TEMA

- #spresnenie-temy - kanál na spresnenie témy.
- #linky-k-teme - linky ktoré sa priamo týkajú témy alebo *state-of-the-art* postupov.

E-mail tímu (deepsearch26@gmail.com) je určený na komunikáciu s osobami mimo tímu. Správy písané z tímového e-mailu reprezentujú názor všetkých členov tímu a preto musí byť obsah týchto správ poskytnutý všetkým členom tímu na posúdenie, a prípadné vyjadrenie pripomienok.

Správou účtu a prijímaním či odosielaním tímových správ sú poverení *scrum master* a manažér komunikácie.

6.3. Zdieľanie súborov

Pri komunikácii je potrebné aj zdieľanie súborov. V tíme využívame zdieľané Google disky pre tímový projekt, môžeme efektívne pracovať na jednom dokumente simultánne. V prípade, že je potrebné vytvoriť nový dokument, člen tímu ho jednoducho vytvorí a odkaz k nemu umiestni do kanálu #*linky-doc* s potrebným popisom.

6.4. Osobná komunikácia

Komunikácia osobne je najhodnotnejšou formou komunikácie a preto je potrebné dodržiavať základné pravidlá slušnosti. Pre produktívny, spravodlivý a slušný priebeh diskusie je niekedy potrebné ju riadiť. Túto úlohu má manažér komunikácie.

7. Metodika pre tímové stretnutia

Táto metodika sa zaoberá plánovaním tímových stretnutí a priebehom stretnutí. Taktiež zahrňuje spôsob zapisovania poznámok počas stretnutí.

7.1. Termín a miesto stretnutia

Pravidelné stretnutia sa uskutočňujú každý v pondelok od 11:00 do 14:00 na FIIT STU v miestnosti 3.28. Je veľmi dôležité, aby sa tohto stretnutia zúčastnili všetci členovia tímu, nakoľko sa tu dohaduje väčšina dôležitých informácií o priebehu vývoja a plánovania.

Sekundárny termín stretnutí nie je presne určený. Je ale efektívne uskutočniť medzistretnutie, ktoré sa uskutočňuje vo štvrtok o 19:00 na FIIT STU v miestnosti -1.31. Tieto stretnutia sú oznámené prostredníctvom komunikačného kanála. Týchto stretnutí sa nemusia zúčastniť všetci členovia tímu, i keď je to veľmi užitočné a podporované. Ak je potrebné, aby niektorý z členov na takomto stretnutí bol zúčastnený, je to dopredu zadefinované a oznámené prostredníctvom komunikačného kanála.

Vzhľadom na rôznorodosť domácej lokácie členov tímu nie je vždy vhodné uskutočniť osobné stretnutia. V prípade potreby je možné vykonať spontánny online hovor (konferenčný hovor) prostredníctvom komunikačného kanála. Tieto hovory nemusia byť oznámené predom. Ich frekvencia nie je definovaná. Takýchto stretnutí sa nemusia zúčastniť všetci členovia tímu.

7.2. Priebeh a obsah stretnutí

Stretnutia budú vedené *scrum-masterom* (Peter Berta). Ostatní účastníci stretnutia nebudú vedúcemu stretnutia skákať do rečí, ale počkajú, kým im dá slovo.

Na pravidelných stretnutiach v pondelok bude prioritnou agendou plánovanie šprintov, ich začínanie a ukončenie. Budú sa tu vykonávať retrospektívy, analyzovať vykonané úlohy, ich kvalita či korektnosť.

7.3. Zápisky zo stretnutí

Ako manažér dokumentácie, zapisovanie poznámok zo stretnutí má na starosti Bronislava Pečíková. To však neznamená, že bude vždy robiť poznámky ona. Znamená to, že ona bude zodpovedná za to, aby aspoň jeden člen tímu na stretnutí robil poznámky z preberanej tematiky.

Poznámky zo stretnutia sú následne spracované a vložené do zápisnice zo stretnutia. Je dôležité, aby tieto poznámky jasne vyjadrovali dohodnuté veci na stretnutí.

Podrobnejšie informácie o zapisovaní stretnutí sú uvedené v metodike *Metodika pre písanie dokumentov*.

7.4. Výnimky (zmena plánu)

V prípade, že sa zmenia podrobnosti ohľadom plánovaného stretnutia (miesto, čas atď.) alebo sa niekto stretnutia nemôže zúčastniť, je potrebné o tom oboznámiť ostatných členov tímu prostredníctvom komunikačného nástroja.

8. Metodiky pre písanie dokumentácie

8.1. Metodika pre zápisnice zo stretnutí tímu

Po každom stretnutí tímu je potrebné vypracovať zápisnicu, ktorá má nasledovnú štruktúru:

X. stretnutie

Dátum: dd.mm.yyyy

Začiatok stretnutia: hh:mm

Koniec stretnutia: hh:mm

Miesto: Budova a číslo miestnosti

Zapísal: Meno Priezvisko

Zúčastnení:

Meno Priezvisko

Meno Priezvisko

Meno Priezvisko

...

Neprítomní:

Meno Priezvisko

...

Obsah stretnutia:

Opis priebehu stretnutia

Závery vyplývajúce zo stretnutia:

- záver č.1
- záver č.2
- ...

Úlohy vyplývajúce zo stretnutia:

- úloha č.1 – zodpovedný
- úloha č.2
- **...**

Pričom význam jednotlivých polí je nasledovný:

- “Zúčastnení” - zoznam členov tímu, ktorí sa zúčastnili stretnutia, v prípade, že bol niektorí z členov tímu prítomný prostredníctvom hlasového kanálu aplikácie Discord je pri jeho mene uvedená poznámka “(Discord)”.
- “Neprítomní” - zoznam členov tímu, ktorí sa stretnutia nezúčastnili.
- “Obsah stretnutia” - opisuje priebeh stretnutia, stručne a v odrážkach, pričom jednotlivé body by mali začínať frázami ako sú:
    - “diskutovali sme….”
    - “dohodli sme sa…..”
    - “identifikovali sme……”
- “Závery vyplývajúce zo stretnutia” - v tejto časti sú uvedené závery na ktorých sme sa ako tím počas stretnutia zhodli. Môže to byť napríklad výber technológie, zavedenie novej metodiky alebo návrh nejakého komponentu systému.
- “Úlohy vyplývajúce zo stretnutia”- v tejto časti sú vymenované úlohy, ktoré sme počas stretnutia identifikovali, k úlohe môže ale nemusí byť pridelený zodpovedný. V prípade, že úloha má prideleného zodpovedného ten je uvedený hneď za úlohou.

Zodpovedný za zápisnice zo stretnutí je manažér dokumentácie, tento musí buď zápisnicu sám vytvoriť alebo tým poveriť jedného z členov tímu. Pred každým stretnutím tímu je potrebné v dokumente *StretnutiaTimu* vytvoriť hore uvedenú šablónu zápisnice. Obsah zápisnice sa tvorí počas stretnutia, tak aby mal každý z členov tímu možnosť zápisnicu sledovať. V prípade, že má niektorý z členov tímu výhrady voči obsahu zápisnice je potrebné ešte na stretnutí tieto výhrady prediskutovať a zápisnicu upraviť. Autor zápisnice vykoná ešte v deň stretnutia finálne úpravy dokumentu (jedná sa najmä o úpravy formulácie textu, ktorý vznikol počas stretnutia, prípadne kontrolu gramatiky) a takto upravený dokument uloží na Google Drive, kde majú ostatní členovia tímu opäť možnosť daný dokument zrevidovať a na prípadné nepresnosti upozorniť prostredníctvom komentára. Týmto postupom eliminujeme problém neúplných alebo subjektívnych zápisníc. Za následné zverejnenie zápisnice na webovom sídle tímu je zodpovedný manažér komunikácie, tento môže štruktúru zápisnice upraviť tak aby bola dostatočne reprezentatívna, pričom obsah zápisnice musí zostať nezmenení.

8.2. Metodika pre dokumentáciu šprintu

Za dokumentáciu šprintov je zodpovedný manažér dokumentácie. Po skončení každého šprintu je potrebné vyexportovať *BurnDown* grafy a tabuľky úloh zo *ScrumDesc*-u, tieto sa následne uložia do dokumentu *ExportEvidencieUloh* v zdieľanom priečinku na Google Drive. Okrem aktualizácie dokumentu *ExportEvidencieUloh* je potrebné aktualizovať kapitolu Sumarizácie šprintov v dokumente *DokumentaciaRiadenia*. Záznam o šprinte musí byť v nasledujúcom tvare:

X.Šprint

Dátum začiatku: dd.mm.yyyy

Dátum konca: dd.mm.yyyy

Ciele šprintu

Zhodnotenie

ScrumDesc Šprint Report

- Šprint Overview
- Prehľad podľa hodín
- Burndown Chart
- Šprint Backlog
- Release Burndown

Retrospektíva

Na začiatku sú uvedené dátumy odkedy a dokedy šprint trval, nasleduje opis šprintu, ktorý pozostáva z uvedenia hlavných cieľov daného šprintu a zhodnotenia šprintu. Po zhodnotení nasledujú reporty grafov a tabuliek z aplikácie *ScrumDesc*, pod grafmi sa môže nachádzať krátky opis, interpretácia alebo zdôvodnenie grafu. A napokon je uvedená retrospektíva, ktorá pozostáva z troch bodov:

- Pozitíva - pozitíva šprintu
- Negatíva - aká problémy sme v tomto šprinte identifikovali
- Návrhy na zmenu - opatrenia, ktoré zavedieme, aby sme sa identifikovaným problémom v budúcnosti vyhli

8.3. Metodika pre dokumentáciu inžinierskeho diela

Je nevyhnutné počas implementácie nejakej súčiastky hneď tvoriť aj dokumentáciu, tá sa zapracováva do dokumentácie inžinierskeho diela, ktorá sa nachádza v zdieľanom priečinku na *google doc* (Tímový Projekt / Dokumentácia / ProjektovaDokumentacia) a má nasledovnú štruktúru:

- Úvod
- Globálne ciele projektu
- Celkový pohľad na systém
- Modul č.1
    - Analýza
    - Návrh
    - Implementácia
    - Testovanie
- Modul č.2
    - Analýza
    - Návrh
    - Implementácia
    - Testovanie
- atď …
- Príručky
- Technická dokumentácia

Pred implementáciou modulu je potrebné napísať analýzu a návrh následne si tieto kapitoly prečítajú všetci členovia tímu a zákazník. V prípade, že nemajú žiadne výhrady začne sa implementácia počas ktorej vzniká dokumentácia Implementácie a Testovania.

Po ukončení implementácie a zdokumentovaní súčiastky nasleduje *code review*, súbežne s ktorým by mala prebiehať aj *review* dokumentácie inžinierskeho diela. Je potrebné zrevidovať či bola implementovaná časť dostatočne zrozumiteľne zapracovaná do dokumentácie inžinierskeho diela. Autor nezrevidovanú časť dokumentácie označí žltým pozadím. Ak *reviewer* nájde nejaké nedostatky v dokumentácií inžinierskeho diela, identifikované nedostatky vyznačí v dokumente prostredníctvom poznámky a upozorní na to autora. Po oprave všetkých nedostatkov sa časť dokumentu označí, ako zrevidovaná prostredníctvom odstránenia žltého pozadia.

9. Metodika riešenia úloh

Táto metodika sa zaoberá riešením úloh v rámci projektu. Je tu načrtnutá štruktúra úloh podľa *Scrum-desku*. Následne je tu aj popísaný postup riešenia problémov a uzatváranie úloh.

9.1. Štruktúra úloh

Štruktúra úloh v projekte priamo súvisí so štruktúrou úloh v agilnom vývoji softvéru – v *scrume*. Úlohy sú rozdelené to používateľských príbehov (*User story*) a tie sú zas rozdelené do *epic*-ov. Ilustračný príklad štruktúry úlohy:

- Epic
    - Používateľský príbeh (*User story*)
        - Popis
        - Parametre (nie všetky sú povinné):
            - vstupy, výstupy, ciele, úsilie (*effort*), *story point*
        - Akceptačné kritériá
        - Úloha 1
            - Popis
        - Úloha 2
            - Popis

9.2. Inicializácia úloh

Úlohy sú primárne inicializované na začiatku šprintu. Postup zadávania úloh je nasledovný (pozn.: poradie úkonov nie je striktne definované):

- Vytvorenie používateľského príbehu (*User story*)
  Vytvorenie používateľského príbehu je zväčša inicializované na začiatku šprintu na spoločnom stretnutí. V prípade potreby je možné používateľský príbeh vytvoriť aj mimo týchto stretnutí, je potrebné o tom ale oboznámiť tím prostredníctvom komunikačného kanálu, alebo osobne.

- Bližšie popísanie používateľského príbehu
  Tento úkon zahŕňa vytvorenie stručného popisu používateľského príbehu či krátkeho komentára k jeho priebehu. Taktiež tu môžu byť zadefinované aj vstupy, výstupy, ciele alebo aj iné parametre podľa potreby.

- Zadefinovanie akceptačných kritérií
  Akceptačné kritériá sú definované vlastníkom produktu. Tieto kritériá sú prediskutované s prítomnými členmi tímu a následne je dohodnutá ich finálna verzia.

- Plánovací poker (*Planning poker*)
  Pomocou techniky plánovacieho pokru je možné ohodnotiť rôzne aspekty používateľských príbehov. Vzhľadom na to, že niektoré používateľské príbehy nie je možné ohodnotiť (vyplýva to s povahy používateľského príbehu), aplikácia plánovacieho pokru nie je podmienkou, ale skôr pomôckou.

- Vytváranie úloh

Podľa charakteru používateľského príbehu sú úlohy v rámci neho vytvárané bezprostredne po jeho definícii, alebo neskôr v priebehu šprintu. Úlohy je vhodné vytvárať na spoločných stretnutiach. V inom prípade je potrebné o tom oboznámiť členov tímu prostredníctvom komunikačného kanálu, alebo osobne.

- Odhad časovej náročnosti úloh
  Pre každú úlohu je nutné, aby obsahovala odhad časovej náročnosti. Tento odhad sa robí bezprostredne po vytvorení úlohy. Vo výnimočných prípadoch je možné odhad vynechať. V takom prípade je potrebné prediskutovať situáciu s ostatnými členmi tímu.

9.3. Prideľovanie úloh

Prideľovanie jednotlivých úloh je vykonávané na začiatku šprintu, ideálne hneď po ich vytvorení. Úlohy sú prideľované voľne, teda ak má niektorý z členov záujem o vykonanie niektorej úlohy, je mu pridelená. Nie všetky úlohy však môžu byť vykonané len jedným človekom, vtedy sú k úlohe pridelení dvaja ľudia, alebo jeden zodpovedný, ktorý už úlohu rozdistribuuje medzi ostatných členov. Ak existujú úlohy, ktoré je potrebné vykonať, ale nikto sa k nim nehlási, *scrum--master* (Peter Berta) rozhodne o ich pridelení.

Na stretnutí je vykonávaná snaha, aby boli úlohy rozdelené rovnomerne medzi všetkých členov tímu. Taktiež sa prihliada na to, aby boli úlohy pridelené ľuďom, ktorí majú v danej problematike skúsenosti, alebo je táto úloha nejakým spôsobom spojená s ich rolou v tíme.

9.4. Ukončovanie úloh

Menej podstatné úlohy môžu byť ukončené aj členmi tímu, po dohode s aspoň jedným iným členom tímu. Ostatné úlohy je potrebné ukončovať spolu s vlastníkom produktu na základe akceptačných kritérií. V prípade, že daný výsledok nespĺňa akceptačné kritériá, tento používateľský príbeh je buď doplnený o chýbajúce časti, alebo ak to nie je možné, tak prenesený do ďalšieho šprintu.

Po dohode s väčšinou členov tímu je možné používateľský príbeh (alebo úlohu) aj zrušiť. Takéto konanie sa ale neodporúča.