SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

Metodika pre testovanie aplikácií vyvíjaných metódou Scrum TDD pomocou nástroja Cucumber

Autor: Kazimír Jaroszewicz

Ak. rok: 2011/2012

Obsah

1. Úvod

Táto metodika popisuje proces testovania softvéru ako súčasť životného cyklu programu a takisto bližšie popisuje testovanie softvéru, ktorý je vyvíjaný metódou Scrum. Bližšie vysvetľuje testovanie aplikácií implementovaných v jazyku *Ruby on Rails* pomocou nástroja *Cucumber* .

2. Pojmy

Tabuľka č. 1: Vysvetlenie použitých pojmov

Pojem	**Vysvetlenie**
Agile	Agile sú postupy vývoja produktov, ktoré pomáhajú zefektívniť vývoj vďaka intenzívnej spolupráci tímu a zákazníka
Scrum	Agilný prístup vývoja softvéru. Vývoj prebieha postupne v dopredu určených časových intervaloch (šprintoch) a na konci každého z nich je fungujúca aplikácia
TDD (test driven development)	Vývoj riadený testami, pri používaní tejto techniky sú testy prírastkovo zapisované ešte predtým ako je implementovaný vlastný výkonný kód programu
Scrum master	Úloha v scrum tíme zabezpečujúca smerovanie vývoja tak, aby sa splnili dopredu určené ciele
Backlog	Zoznam požiadaviek na systém
Vlastník produktu	Vytvára backlog systému
Testovací inžinier	Vykonáva testy
Scenár	Slovný popis toho čo Vlastník produktu očakáva od funkcionality danej časti systému
Unit testy	Testovanie jednotiek systému, ktoré predstavujú najmenšiu možnú testovateľnú

	časť systému
Ruby on Rails	Programovací jazyk pre vývoj webových aplikácií
Cucumber	Nástroj pre unit testovanie aplikácií implementovaných v jazyku Ruby on Rails

3. Zoznam nadväzujúcich metodík

Metodika pre tvorbu backlogu projektu riešeného Scrumom

Metodika pre manažment chýb

4. Použitá literatúra a zdroje

[1] Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice, Second Edition, Vyd. 2003, ISBN 0-321-15495-9

5. Manažment testovania

Testovanie ako jedna zo základných disciplín merania kvality je systematická činnosť a má vlastné overené metodické postupy, ktoré správnym premietnutím požiadaviek do testovania zabezpečia kontinuálnu kontrolu kvality. Táto kapitola sa zaoberá procesmi v testovaní softvéru vyvíjaného metódou Scrum TDD.

5.1 Role a zodpovednosti

Tabuľka č. 2: Role a zodpovednosti

Rola	**Zodpovednosť**
Vlastník produktu	• Vytvára backlog systému • Zadáva prioritu jednotlivým požiadavkám

	• Vytvára scenáre pre jednotlivé požiadavky • Vytvára backlog šprintu výberom úloh z backlogu systému
Vývojový tím	• Spolupracuje na návrhu systému podľa backlogu • Zabezpečuje naprogramovanie systému podľa návrhu • Opravuje hlásené chyby systému vyplývajúce z testovania
Scrum master	• Sleduje napĺňanie cieľov • Usmerňuje vývoj vzhľadom na definované ciele • Vytvára akceptačné testy • Ukončuje šprint po naplnení backlogu a úspešných akceptačných testoch
Testovací inžinier	• Vykonáva testy • Vyhodnocuje testy • Podáva hlásenia o chybách

5.2 Proces testovania

Nasledujúca kapitola objasňuje proces testovania softvéru v metóde vývoja Scrum. Definuje jednotlivé kroky v procese testovania a objasňuje ich význam a zodpovednosti, tak ako to je znázornené v tabuľke č. 3.

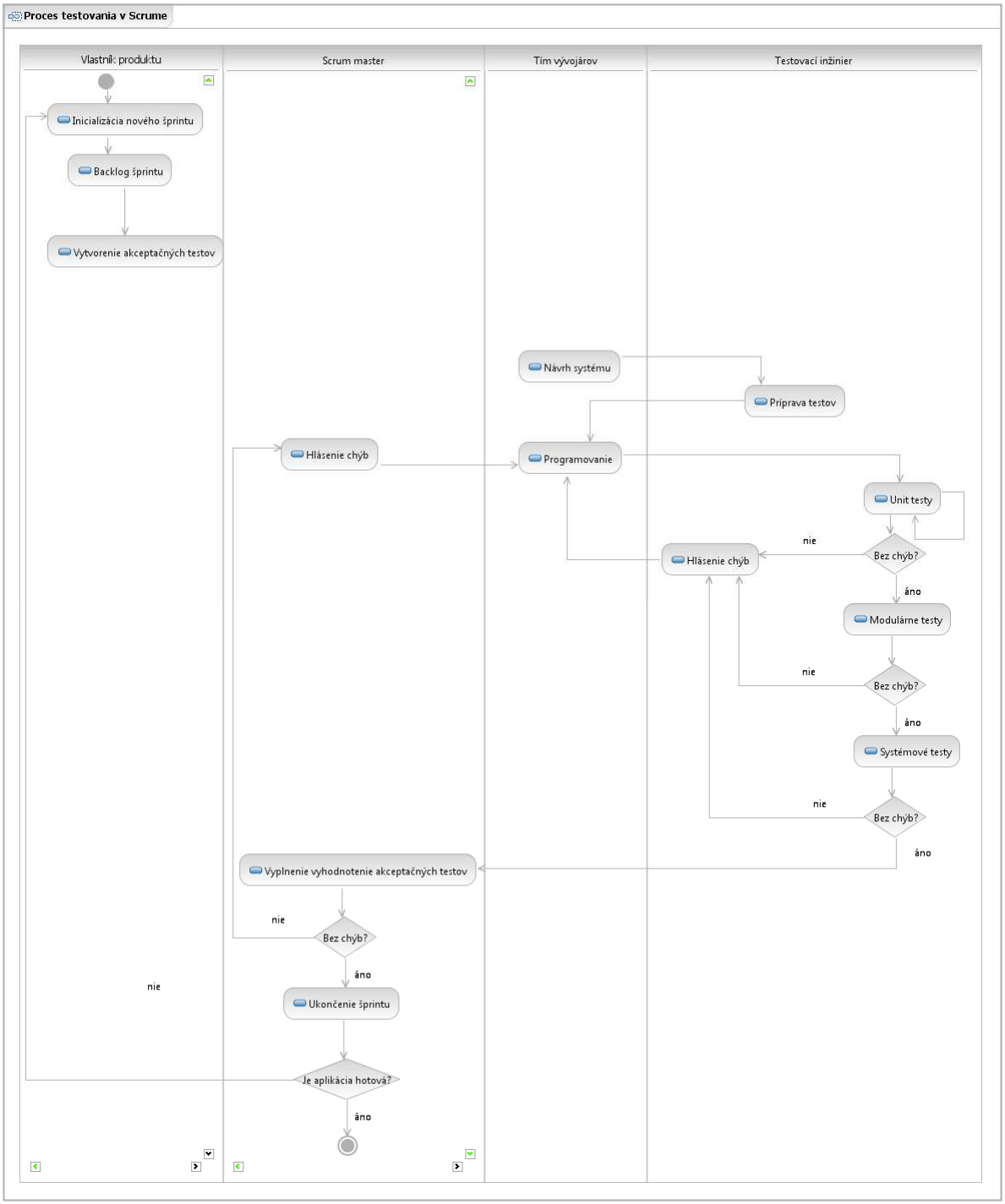
Tabuľka č. 3: Kroky a zodpovednosti v procese testovania

	Krok	**Rola**	**Kapitola**
1.	Špecifikácia požiadaviek – backlog šprintu	Vlastník produktu	5.2.1
2.	Vytvorenie akceptačných testov	Vlastník produktu, Scrum master	5.2.2
3.	Návrh systému	Vývojový tím	5.2.3
4.	Príprava testov	Testovací inžinier	5.2.4
5.	Programovanie	Vývojový tím	5.2.5
6.	Testovanie	Testovací inžinier	5.2.6
7.	Hlásenie o chybe	Testovací inžinier	5.2.7

8.	Vyplnenie a vyhodnotenie akceptačných testov	Scrum master	5.2.8

Obrázok č. 1 znázorňuje diagram aktivít v procese testovania. Je tu priblížená časová následnosť jednotlivých krokov a takisto poukazuje na opakovanie testov v každom šprinte vývoja až po úspešné ukončenie projektu.

Obrázok č. 1: Diagram procesu testovania



5.2.1 Špecifikácia požiadaviek – backlog šprintu

Vstup: *Backlog systému*

Výstup: *Backlog šprintu*

Zodpovednosť: *Vlastník produktu*

Vlastník produktu špecifikuje požiadavky, ktoré je potrebné naplniť v danom šprinte. Spíše ich do uceleného dokumentu, aby boli dostupné vývojovému tímu. Na základe týchto požiadaviek vytvorí aj backlog tohto šprintu a tým zadá svoju predstavu tímu ako má vyzerať aplikácia po ukončení šprintu. Jednotlivé úlohy šprintu majú svoju prioritu, ktorá predstavuje ich dôležitosť pre zákazníka. Úlohy backlogu majú predpísanú formu:

[ID][Názov][Priorita][Náročnosť úlohy][Pod úlohy / úlohu, ktorej sú pod úlohou][Odhadovaný čas][Popis]

Náročnosť úlohy: Tím vývojárov určí obtiažnosť úlohy na základe svojich odhadov a predošlých skúseností

Odhadovaný čas: vypĺňa zodpovedný vývojár a predstavuje odhad času potrebný na splnenie úlohy

5.2.2 Vytvorenie akceptačných testov

Vstup: *Backlog šprintu*

Výstup: *Akceptačné testy k jednotlivým úlohám*

Zodpovednosť: *Scrum master, projektový vlastník*

Vlastník produktu v spolupráci so Scrum mastrom, pre každú úlohu zo šprintu vytvorí akceptačný test (v Scrume *scenár*). Test pozostáva z :

- identifikácie úlohy pomocou *ID* a *názvu*
- *priority* splnenia funkcionality zadanej od produktového vlastníka
- určenia *rozhrania* poskytujúceho interakciu so systémom

- *cieľa*, ktorý daná funkcionalita napĺňa
- *vstupných a výstupných podmienok*
- jednotlivých *krokov*, ktoré vedú k zabezpečeniu cieľa
- *očakávaných reakcií systému*
- *skutočných reakcií systému*, táto kolónka sa vyplní až pri vyhodnotení testov

Tabuľka č. 4: Akceptačný test

ID	x	**Názov**	...	
Úroveň splnenia testu		Musí – ~~Mal by~~ – ~~Mohol by~~		
Rozhranie		...		
Cieľ		...		
Vstupné podmienky		...		
Výstupné podmienky		...		
Krok		**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1		...	...	...
2		...	...	...
...		...	...	...
N		..	..	...

5.2.3 Návrh systému

Vstup: *Backlog šprintu a scenáre*

Výstup: *Návrh systému*

Zodpovednosť: *Vývojový tím*

Vývojový tím na základe požiadaviek pre daný šprint a poskytnutých scenárov vytvorí návrh systému. Keďže sa postupuje metódou Scrum je vhodné si uvedomiť, že v nasledujúcom šprinte sa bude vytvárať nový návrh zohľadňujúci už naprogramované a skompletizované časti. A takto sa postupuje kým nie je aplikácia úspešne dokončená. Táto

časť priamo nesúvisí s problematikou testovania preto sa odkazujem na dokument bližšie opisujúci tento problém [1].

5.2.4 Príprava testov

Vstup: *Akceptačné testy*

Výstup: *Unit, modulárne a systémové testy*

Zodpovednosť: *Testovací inžinier*

Testovací inžinier na základe poskytnutých akceptačných testov vytvorí potrebné testy, ktoré sa po naprogramovaní jednotlivých častí systému spustia a overia funkčnosť systému vzhľadom na poskytnuté akceptačné testy. Postup pre prípravu *Unit testu* je v kapitole 6.

5.2.5 Programovanie

Vstup: *Návrh systému / hlásenie o chybe*

Výstup: *Kód aplikácie*

Zodpovednosť: *Vývojový tím*

Vývojový tím najprv naprogramuje časti systému podľa požiadaviek. Po hlásení o chybe prebehne oprava chyby a odovzdanie programu na pretestovanie.

5.2.6 Testovanie

Vstup: *Kód aplikácie, pripravené testy*

Výstup: *Hlásenie*

Zodpovednosť: *Testovací inžinier*

Testovací inžinier vykoná test dodaného zdrojového kódu. Po nastaní chyby vypracuje hlásenie o chybe. Nulový počet chýb sa považuje za splnenie požiadaviek implementácie a proces testovania končí.

5.2.7 Hlásenie o chybe

Vstup: *Chyba*

Výstup: *Hlásenie chyby vývojovému tímu*

Zodpovednosť: *Testovací inžinier*

Testovací inžinier po identifikácií chyby vypracuje správu o chybe. Správu odovzdá vývojovému tímu , ktorý chybu opraví a posunie kód na pretestovanie.

5.2.8 Vyplnenie a vyhodnotenie akceptačných testov

Vstup: *Aplikácia, akceptačné testy*

Výstup: *Hlásenie chyby vývojovému tímu*

Zodpovednosť: *Scrum master*

Scrum master simuluje používanie aplikácie nastavovaním parametrov systému a klikaním presne podľa akceptačných testov. Vyplňuje kolónku v akceptačných testoch s názvom *Skutočná reakcia*. Následne porovná kolónky *Očakávaná* a *Skutočná reakcia.* Zhoda týchto reakcii znamená úspešné ukončenie šprintu . Nezhoda medzi týmito reakciami spôsobuje odovzdanie hlásenia o chybe vývojovému tímu.

6. Testovanie použitím nástroja Cucumber

Táto časť metodiky podrobne rozoberá *unit testovanie* softvéru implementovaného v jazyku *Ruby on Rails*, použitím nástroja *Cucumber*. Je tu podrobne rozobraná problematika prípravy testu popísaná v kapitole 4.2.4 a takisto aj problematika vykonávania testu z kapitoly 4.2.6. Výsledkom je metodika pre testovanie.

Tabuľka č. 5: Kroky pre testovanie softvéru v nástroji Cucumber

	Krok	**Kapitola**
1.	Vytvorenie opisov a scenárov	6.1
2.	Vytvorenie krokov testu pre scenáre	6.2
3.	Spustenie testu	6.3
4.	Vyhodnotenie testu	6.4

6.1 Vytvorenie opisov a scenárov

Opisy a scenáre pre požiadavku dodáva vlastník produktu. Majú určenú šablónu. Šablóny sú v tabuľke č. 6 a obsahujú aj anglickú (pôvodnú) verziu. Tieto scenáre sa ukladajú do priečinku *názov_aplikácie/features* a súboru *názov_požiadavky.features.*

Tabuľka č. 6: Šablóny pre opis a scenár úlohy

Opis	*Názov požiadavky* *Ako [používateľská rola] chcem[požiadavka]aby [účel]* *As[role]I want[feature]so that[benefit]*
Scenár	*Ak [podmienka]* *A [ďalšia podmienka]* *Keď [akcia používateľa]* *A[ďalšia akcia]*

	Potom [želaný stav] *A [ďalší želaný stav]*
	Given [some precondition] *And [some other precondition]* *When [some action by the actor]* *And [some other action]* *And [yet another action]* *Then[some testable outcome is achieved]* *And [something else we can check happens too]*

Scenár je napísaný v jazyku *Gherkin.* Tento definuje slová ako *Given, When, And, Then.* Podmienka , akcia používateľa a želaný stav obsahujú slová napísané v úvodzovkách, ktoré predstavujú premenné.

Príklad:

Scenario: Seeing dashboard after login

Given I am logged out

When I log in

Then I should be on my dashboard page

6.2 Vytvorenie krokov testu pre scenáre

Definovaním krokov pre daný scenár vytvoríme pre *Cucumber* postupnosť ako sa má preklikať aplikáciou a ako má vyplňovať údaje. Kroky sa ukladajú do priečinku *názov_aplikácie/features/steps_definition* a súboru *názov_požiadavky*_steps.rb. Spustením testu bez definovania krokov *Cucumber* sám navrhne šablónu pre ich definovanie. *Cucumber* test spustíme príkazom:

cucumber features/názov_požiadavky.features

Pre každý z riadkov scenáru (krokov) vytvoríme blok kódu v jazyku *Ruby:*

- Odpíšeme kľúčové slovo jazyka *Gherkin*
- Kľúčové slovo *And* nahradíme kľúčovým slovom ***Given***, ***When*** alebo ***Then***, podľa toho, ktoré je v scenári nad ním
- Medzi znaky **/^** a **$/** vložíme riadok scenáru
- Text v úvodzovkách
- **do** (rob)
- **|*názov_premennej*|** (počet sa rovná počtu slov v úvodzovkách)
- Nový riadok
- Ruby kód implementujúci riadok popísaný v predchádzajúcich bodoch

Príklad:

```
When /^I log in$/ do
  @current_user = Factory(:user)
  visit destroy_user_session_path
  visit new_user_session_path
  fill_in "Email", :with => @current_user.email
  fill_in "Email", :with => @current_user.email
  fill_in "Password", :with => @current_user.password
  click_button 'Sign in'
end
```

6.3 Spustenie testu

Do konzoly napíšeme príkaz pre spustenie testu:

```
Cucumber features/názov_požiadavky.features
```

6.4 Vyhodnotenie testu

Tabuľka č. 7: Vyhodnotenie testov

Chyba	• Zvýraznenie červenou farbou • Chybová hláška bližšie špecifikujúca dôvody neakceptovania implementácie
Bez chyby	• Zvýraznené zelenou farbou • Test prebehol úspešne