2017


# Is TDD a form of OCD?

4SA432 – SEMESTRÁLNÍ PRÁCE

**MICHAEL IRIŠEK** (IRIM00), **MURAT SEIDOV** (SEIM00), **JURAJ OCHTINSKÝ** (XOCHJ00)

ZIMNÍ SEMESTR 2017/2018, ODEVZDÁNÍ DO 17.12.2017 23:59

# OBSAH

# 1. Úvod

Na metodiky vývoje softwaru je často nahlíženo pouze z několika základních úhlů. Řeší se, zda zefektivňují práci, umožňují se vyhnout chybám, nebo třeba lépe spojují vývojářský tým. Všechny tyto pohledy na věc jsou samozřejmě důležité, ale existuje další velká spousta obvykle ignorovaných souvislostí, jejichž dopady mohou být někdy stejně zásadní. Tato práce se jedné z těchto na první pohled nejasných souvislostí věnuje a dále ji rozvádí.

Jejím hlavním účelem je představit propojení metodiky vývoje softwaru TDD se vznikem a podporou obsesivně kompulzivních poruch. Snaží se tak neinformovat jen o již známých nedostatcích této a podobných metodik, ale i o jejích dalších negativních dopadech. Práce se proto snaží informovat čtenáře o základní problematice vývoje řízeného testy a obsesivně kompulzivních poruch, ukazuje možné spojení těchto dvou termínů a snaží se nalézt vhodné řešení zkoumaného problému.

## 1.1 Cíle práce

Práce si klade za cíl seznámit čtenáře se základními poznatky z odborného článku „Is TDD a form of OCD?“ a rozšířit je o dodatečné informace pro jejich podrobnější analýzu a pochopení. Na počátku je věnován čas teorii o Test Driven Development (TDD) a obsesivně kompulzivních poruchách (OCD). Následně jsou tyto dva pojmy vloženy do kontextu článku a jsou zkoumány vyvozené závěry z něj plynoucí. Čtenář je poté informován, jak mohou být zkoumané problémy řešeny, jakou úlohu v problematice hraje Behavioral Driven Development (BDD), který je taktéž popsán, a jaká je jeho očekávatelná budoucnost.

## 1.2 Struktura práce

Semestrální práce je rozdělena na dvě hlavní části. První definuje a popisuje vývoj řízený testy a teorii týkající se obsesivně kompulzivních poruch. Následně tyto dva pojmy vkládá do souvislosti a seznamuje se závěry, které byly prezentovány v odborném článku, jímž je tato práce volně inspirována.

Druhá část práce se věnuje možnému řešení představených problémů a konkrétně přibližuje jedno z možných, a sice vývoj řízený chováním.

Závěr shrnuje poznatky zjištěné během psaní práce.

# 2. Problémy s TDD a spojení s OCD

## 2.1 Test Driven Development (TDD)

Test Driven Development (dále zkracováno jako „TDD“ a do češtiny překládáno jako „Vývoj řízený testy“) je metodika vývoje softwaru obsahující soubor postupů, jejichž hlavním účelem je agilní produkce odladěného a funkčního kódu. (Agiledata.org, 2017) Jak už název napovídá, klíčovými jsou v tomto druhu vývoje především testy, ze kterých následně vychází i samotný kód.

Zjednodušeně řečeno se v rámci TDD opakuje několik dílčích kroků, jejichž výsledkem je finální verze kódu. Zaprvé je nutné stanovit úkol, který program, nebo jeho část, bude řešit. Klíčová je velikost takovéhoto úkolu, která by měla být vždy co nejmenší možná. Testování i programování komplikovaných a dlouhých úkolů by v opačném případě bylo přespříliš složité. Samozřejmě je nutné stanovit takové úkoly, které vůbec otestovat lze.

Následuje stanovení všech potřebných testů, které po dokončení programátorské práce zkontrolují správnost a funkčnost nového kódu. Testy musí kontrolovat veškeré eventuální problémy, které by se v kódu mohly vyskytovat. Právě na tomto kroku závisí úspěch TDD metodiky. Bez správně napsaných a kompletních testů nemůže být zvolený úkol správně naprogramován. Samozřejmé je, že testy jsou přidávány i při samotném programování, pokud vyplynou z povahy právě tvořeného kódu, nebo byly na počátku opomenuty. Před samotným programováním jsou testy spuštěny a je ověřeno, že všechny končí chybou. Takové testy, pro které toto pravidlo neplatí, jsou buď špatně navržené (je nutné změnit jejich logická pravidla), nebo obsahují chyby.

Následuje tvorba kódu, při které jsou vytvořené testy využívány pro ověření funkčnosti nově vytvořených funkcionalit programu. Nejdůležitějším je ovšem poslední krok, a sice úspěšné proběhnutí všech testů po dopsání kódu. V případě dobře navržených testů absence jakýchkoliv chyb dokazuje správnou funkčnost nově implementované funkcionality. I přesto není kód ještě finální. V praxi je běžné, že je kód potřeba upravit, optimalizovat a „vyčistit“ – jeho refactoring je proto v rámci TDD taktéž zohledněn a platí, že i po jeho proběhnutí musí stále všechny testy bez problémů projít. V některých případech je možné několik testů i přidat, pokud například v rámci refactoringu probíhá i určitá optimalizace zrychlující běh programu. (Agiledata.org, 2017)

Tento popis TDD je pochopitelně jen zjednodušený a celá metodika obsahuje dodatečné postupy a nástroje na efektivnější vývoj.

Teoretických výhod tohoto přístupu je hned několik. První je očividná výhoda rychlé tvorby stoprocentně funkčního a čistého kódu, který lze ihned implementovat. Druhým pozitivem je schopnost kód rychle aktualizovat a rozšiřovat podle potřeby.

Spolu s výhodami TDD existuje nicméně i množství problémů a nevýhod tohoto přístupu, kterým se věnuje jedna z následujících kapitol. (Schoots, 2014)

Článek, ze kterého tato semestrální práce vychází však formuluje ještě jednu hypotézu, která se nevěnuje nedostatkům TDD pouze u jedince, nýbrž u celého týmu. (Maayan, 2017) Nejprve zmiňuje případ, kdy jsou některé agilní vývojářské týmy zaměřené více na samotné testování než vývoj kódu. Členové týmů se snaží najít co nejvíce chyb v připravovaných nových verzích programů od svých spolupracovníků, což vytváří příliš silný tlak na vývojáře, kteří mohou být za tyto chyby trestáni. Dopad tohoto chování pak nezasahuje pouze jednotlivce a trpí jím celý tým. Kvůli tomuto přístupu se pak u členů týmu mohou dokonce vyvinout poruchy jako „Evaluation Anxiety“ – strach z ohodnocení.

Autor článku pak TDD vnímá jako snahu tomuto problému předejít. Tento přístup ale v jeho očích pouze mění jeden problém za druhý a mnohdy vytváří negativa úplně nová, často hraničící s obsedantně kompulzivními poruchami (OCD). V části věnující se TDD autor dochází k závěru, že vývoj řízený testy vede ke kompulzivnímu a přehnanému testování každé řádky kódu, jen aby se vývojář vyhnul případnému selhání v očích svých týmových spolupracovníků i kvůli těm nejmenším chybám. Pokud se situace nemění, opakuje se a vývojář je vystaven většímu pracovnímu stresu, může vést až k závažným psychickým poruchám a nemocem, které TDD neeliminuje, ale naopak je ještě dále podporuje. Takovéto chování je téměř učebnicovým případem Evaluation Anxiety. (Informed Health Online, 2017)

## 2.2 Obsessive-compulsive Disorder (OCD)

V nedávné době se i mezi širokou veřejnost rozšířily znalosti týkající se zkratky OCD (i v češtině vyslovovaná jako *ou-sí-dý*), tedy o obsedantně kompulzivní poruše. Této neurotické poruše se především kvůli zájmu některých celebrit a slavných osobností dostává velké pozornosti. Důvodem taktéž může být nedávné zjištění, že obsedantně kompulzivní poruchy nejsou mezi běžnými lidmi nijak vzácné. Některé výzkumy dokazují možnou přítomnost této nebo souvisejících neurotických poruch ve 3-5 % populace. Projevovat se mohou rozličnými způsoby. (Informed Health Online, 2017)

Hlavním znakem těchto poruch je přítomnost neustálých nutkání a vtíravých myšlenek, které jsou označovány obsese. Nejen že je pro člověka trpícího OCD přítomnost takovýchto myšlenek krajně nepříjemná, ještě větším problémem je nutnost tato nutkání i přes nevůli plnit.

Obsese se mohou vyskytovat v různých podobách. Časté jsou takzvané „rituály“, které člověk pravidelně provádí – opakovaná kontrola zamknutého automobilu, vypnutého plynu, zavřených dveří apod. Stejně tak jsou obvyklé různé fobie s OCD spojené (opakované mytí rukou kvůli strachu z bakterií), často založené na absurdních předpokladech. Takto nemocní lidé například mají potřebu opakovat stále ta samá slova, v mysli počítat do určitého čísla, než něco udělají apod. (Informed Health Online, 2017)

Výjimkou nejsou ani různé posedlosti – určitými barvami, nutnost schraňovat všechny předměty včetně odpadků, potřeba plánování nastávajícího dne včetně základních potřeb do poslední minuty, hypochondrie apod. Je tedy zřejmé, že mezi obsedantně kompulzivní poruchy spadá velké množství

problémů, které se mohou v budoucnu dále vyvíjet kvůli stresu, osobním problémům, genetickým predispozicím, nebo jiným nemocem.

Pro účely práce je nutné podrobněji popsat jednu z poruch, která rámcově patří mezi OCD. Ačkoliv je pro ni možné v češtině používat název „strach z ohodnocení“, v praxi je běžnější se setkat s termínem v anglické podobě, a sice Evaluation Anxiety. Ohledně tohoto problému panuje spousta mýtů a bohužel je často nepochopen. Výjimkou nejsou například komentáře rodinných příslušníků či přátel, přirovnávající člověka trpícího tímto problémem k dítěti školního věku, obávajícího se špatné známky.

Evaluation Anxiety je ale mnohem závažnějším problémem, který by se navíc mohl týkat téměř každého. Podařilo se prokázat, že ke vzniku této poruchy napomáhá špatné pracovní prostředí, problematická výchova, genetické predispozice, nebo i jen přespřílišný stres a může se tak objevit i u lidí, kteří jsou jinak jak po fyzické, tak po psychické stránce zdraví.

Jak už bylo naznačeno v předchozí kapitole, v naprosté většině případů se Evaluation Anxiety objevuje u zaměstnanců, kteří jsou vystaveni tlaku svého pracovního okolí a nabydou dojmu (ať už mylně, nebo důvodně), že jsou neustále kontrolováni a jejich pracovní výkon není dostatečně dobrý. Člověk, který Evaluation Anxiety trpí se poté snaží práci buď kompletně vyhýbat, popřípadě pokud to není možné, udělá cokoliv, aby případnému negativnímu hodnocení předešel. (Informed Health Online, 2017)

## 2.3 Paralela mezi TDD a OCD

Z předcházející kapitoly vyplývá spojení těchto dvou zmíněných problematik. Hlavní premisou článku, ze kterého tato práce vychází, je existence určitých souvislostí mezi vývojem vedeným testy a obsesivně kompulzivními poruchami. Hlavním argumentem pro podporu této hypotézy je samotná funkčnost TDD, která se už principiálně podobá chování člověka trpícím Evaluation Anxiety.

Autor článku dokonce přiznává, že podle jeho názoru TDD vzniklo vyloženě jako obranný mechanismus vývojářů, jak se vyhnout kritice svého kódu. Jedná se samozřejmě o subjektivní názor, které by bylo těžké potvrdit. Zároveň ale dokazuje, že problém může být ještě mnohem větší, než se na první pohled může zdát. (Maayan, 2017)

Nejdůležitějším poznatkem je, že TDD jako takové nevyvolává zkoumanou Evaluation Anxiety. K její tvorbě přispívají jiné faktory na pracovišti i v osobním životě jednotlivců. Problém nastává až v momentě, kdy se u konkrétního vývojáře nějaká z obsesivních poruch objeví. V tom případě právě TDD může vést k jejímu dalšímu prohlubování a podporování.

Místo tvorby kódu totiž začne docházet ke kompulzivnímu psaní dalších a dalších testů, které mají vývojáři zajistit jeho bezchybnost a dokonalost. Tím si chce zajistit, že nebude později za svou práci kritizován a že se mu nebude věnovat větší pozornost.

Rozsah spojení mezi TDD a Evaluation Anxiety není dodnes nijak dále konkretizován, nebo měřen. Přesto nelze popřít svědectví a názory lidí, kteří se setkávají se stejným chováním (ne-li ho

přímo zažili) v rámci své práce. Problém zde jistě existuje, jen nikdo pořádně neví, jak velký je. Může se jednat jen o drobnou korelaci, nebo lze prvky OCD považovat za přímou součást TDD – k tomuto názoru se i přes jistou kontroverzi kloní autor článku. (Maayan, 2017)

# 3. JAK PROBLÉMY S TDD ŘEŠIT

## 3.1 IDENTIFIKACE HLAVNÍCH PROBLÉMŮ S TDD

I přes zcela logické důvody, proč se TDD může zdát jako vhodná metodika pro spoustu vývojářských týmů, obzvlášť v nedávných letech od ní naopak spousta velkých společností začíná rychle upouštět. Důvodem není jen v této práci prezentovaná spojitost s psychickými problémy, ale především spousta problémů, které se při reálném používání TDD začaly objevovat a mnohdy převažují nad předpokládanými klady. (Schoots, 2014)

Nasnadě je fakt, že ne všichni vývojáři jsou ochotní se tomuto přístupu podřídit. Často argumentují vlastními naučenými postupy s větší efektivitou a stejně kvalitním kódem. Dalším problémem bývá vágní či absolutní nedodržování postupů v TDD v rámci týmů, které ho mají využívat. Běžně se lze setkat s názorem, že v programátorské praxi vývojář nedokáže napsat efektivní testy ještě před tím, než vůbec začne programovat, jelikož řešené úlohy jsou velmi složité, zaberou spoustu času k vyřešení a testy musí být přizpůsobeny až způsobu, kterým je úloha řešena. Právě neefektivita pak bývá zmiňována jako hlavní důvod, proč vývoj řízený testy vůbec nepoužívat, nebo si tuto metodiku alespoň v rámci týmu poupravit a „polidštit“.

Při hlubším průzkumu TDD a jeho aplikaci v praxi, tj. na reálně běžících projektech, lze v metodice vypozorovat několik dalších závažných problémů.

Na začátku každého řešeného problému je určen úkol, od kterého se odvíjí celý zbytek řešení. Podle metodiky následuje sestavení všech potřebných testů, které nikdy nesmí projít (fáze „červená“) a pak tvorba samotného kódu. Až v momentě, kdy kód projde všemi testy (fáze „zelená“) a je optimalizován (fáze „refactoring“) je kontrolován samotný návrh řešeného úkolu na základě jeho funkčnosti v kontextu celého projektu. Pokud je tento návrh špatný, celý proces byl zbytečný a musí se opakovat, což představuje velkou nevýhodu oproti jiným metodikám a znamená další zdržování a zvyšování nákladů.

Nevýhodu mohou být i samotné testy, které představují základ celé metodiky. Jak již bylo naznačeno, někteří vývojáři odmítají psát větší množství testu před začátkem programování (čímž přestávají metodiku následovat), nebo v horším případě píšou naopak testů až přespříliš. Přebytečné množství testů nejenže zpomaluje vývoj, ale neznamená automaticky produkci lepšího kódu, tudíž nezajistí ani vyšší obecnou kvalitu.

Posledním, a možná i nejzávažnějším problémem zmíněným v této práci, je fakt, že v praxi TDD často neposkytuje žádné zlepšení kódu oproti jiným metodikám, jak agilním, tak rigidním. Vytvořený kód tudíž sice není špatný, ale není o nic lepší než kód, který by byl vytvořen pomocí

jiné metodiky, mnohem jednodušší na následování a především rychlejší. Pokud bychom si jako metriku úspěchu nasazení nové vývojové metodiky nastavili produkci kvalitního kódu, TDD by mohlo být hodnoceno jako naprosto nevyhovující. (Schoots, 2014)

Především z těchto důvodů vzniklo několik dalších metodik, které chtějí přijmout všechny dobré věci, které TDD představuje a vylepšit jeho slabiny. Jedním z nejznámějších příkladů takové metodiky je Behavioral Driven Development. (North, 2016)

## 3.2 Behavioral Driven Development (BDD)

Behavioral Driven Development (někdy je i přes lehký gramatický rozdíl používán název „Behavior Driver Development“ a do češtiny je překládán jako „Vývoj řízený chováním“) je metodika pro vývoj softwaru vzniknuvší z TDD. Vývoj řízený testy byl v tomto případě doplněn o principy a další myšlenky z Domain-Driven-Designu pro lepší spolupráci týmu a použití nových nástrojů, které by měly usnadnit vývoj.

Hlavní myšlenkou BDD je zapojení jak manažerské, tak technické části týmu do vývoje. Tento přístup lze nejlépe pochopit z testů, které tato metodika produkuje. Na rozdíl od TDD jsou zaměřené na chování (proto pojmenování „Behavioral“). Toto chování je definováno managementem a je vytvořeno tak, aby vytvářelo budoucí hodnotu v rámci řešeného projektu. Při správném použití je pochopitelné jak pro vývojáře, tak pro již zmíněný management. Tato tvorba hodnoty pro celek je pak v rámci metodiky nazývána „outside-in“ aktivitou.

Při hodnocení dalších nástrojů využívaných v rámci této metodiky je nutné přiznat, že zde je inspirace a vycházení z TDD nejjasnější. BDD používá často stejné, nebo minimálně podobné nástroje, které mají za hlavní úkol automatizovat, podpořit a zjednodušit proces vývoje. (North, 2016)

Specifikace chování, tedy plánovaný řešený úkol, je stále neoddělitelnou a důležitou součástí metodiky. Důležitou novinkou snažící se řešit problémy, které nevhodně stanovené úkoly v rámci TDD mohou mít, je popis chování ve formě příběhu. Ten je vytvořen na základě analýzy a návrhu a má podobu formálního jazyku, ve většině případech v anglickém jazyce. Díky tomuto postupu je nejen řešený problém srozumitelný každému, zároveň zajišťuje jeho vhodnost zvolení.

Způsob popisu chování má určitá pravidla, která byla sestavena na základě logiky Hoare a doménově specifickém jazyku. Může vypadat například takto (Jbehave, 2016):

```
+Scenario 1: Account is in credit+
Given the account is in credit
And the card is valid
And the dispenser contains cash
When the customer requests cash
Then ensure the account is debited
And ensure cash is dispensed
And ensure the card is returned
```

Konkrétním nástrojem používaným v praxi v rámci metodiky BDD se stal například jazyk **Gherkin**, který umožnuje ověřit chování softwaru bez konkrétní specifikace způsobu implementace. Jeho nespornou výhodou je možnost použití i v segmentu s lidmi bez technologického vzdělání kvůli své přirozené syntaxi. Díky tomu je možné do vývoje zapojit i běžné uživatele a urychlit tak komunikaci mezi nimi a vývojáři. V praxi navíc dochází i k lepšímu pochopení požadovaných novinek a změn, takže pozitivní dopad lze sledovat i z hlediska efektivity vývoje. Použití jazyku Gherkin v kombinaci s nástrojem JBehave může v praxi vypadat například následovně (Jbehave, 2016):

```
Given a 5 by 5 game
When I toggle the cell at (3, 2)
Then the grid should look like
.....
.....
.....
..X..
.....
When I toggle the cell at (3, 1)
Then the grid should look like
.....
.....
.....
..X..
..X..
When I toggle the cell at (3, 2)
Then the grid should look like
.....
.....
.....
.....
..X..
```

Následné namapování scénáře do metod v Javě může vypadat takto (Jbehave, 2016):

```
private Game game;
private StringRenderer renderer;

@Given("a $width by $height game")
public void theGameIsRunning(int width, int height)
{
game = new Game(width, height);
```

```
        renderer = new StringRenderer();
        game.setObserver(renderer);
    }
      @When("I toggle the cell at ($column, $row)")
    public void iToggleTheCellAt(int column, int row)
    {
        game.toggleCellAt(column, row);
    }
    @Then("the grid should look like $grid")
    public void theGridShouldLookLike(String grid)
    {
    assertThat(renderer.asString(), equalTo(grid));
    }
```

Ve výsledku představuje Gherkin nástroj, kterým se dá nejen otestovat vytvářený kód, ale i zkontrolovat jeho chování popsané běžným uživatelem. Testy pak nejsou tím nejdůležitějším prvkem metodiky, ale pouze prostředkem pro rychlejší tvorbu srozumitelnějšího kódu, což je cíl, o který od počátku TDD usilovalo a až BDD ho v tomto pohledu dotáhlo k dokonalosti.

Pro kontext této práce je důležité dodat, že právě tato vlastnost BDD, a sice zaměření na chování, má podle autora článku pozitivní dopad na lidi trpící Evaluation Anxiety. Nemíří jen na stálou kontrolu správnosti kódu jednotlivce, která může zmíněnou psychickou poruchu vyvolávat a TDD místo jejího řešení jí ještě dále podporovat. Naopak je BDD zaměřeno na týmovou práci a její hodnocení v rámci společenství lidí. Vývoj řízený testy je tedy vnímán jen jako defenzivní mechanismus pro obhajobu kódu jednotlivce, kdežto vývoj řízený chováním je považován za zdravější přístup. (Maayan, 2017)

Všechny zmíněné výhody samozřejmě platí jen v případě, že je BDD správně dodržováno a nikdo z týmu ho záměrně nesabotuje. Jako všechno ostatní, má Behavioral Driven Development i své chyby.

## 3.3 Prostor ke zlepšení BDD

Největší nedostatek vývoje řízeného chováním tkví v jeho části, která zůstává i v jiných metodikách neměnná po desítky let. Testy jsou v rámci BDD sice pohodlnější, pochopitelnější a opravdu při správném použití dokáží nejen zrychlit vývoj, ale dále ho i zefektivnit a přiblížit běžným uživatelům. Jejich vyhodnocování je ale stále stejné a může představovat závažný problém. (North, 2006)

Pokud jsou řešené úkoly jednoduché a nenáročné, většinou na tento problém nenarazíme. Odborníci nicméně tvrdí, že při vývoji velkých aplikací není možné se zaměřovat jen na kodérskou stránku vývoje. Je nutné brát v úvahu i uživatelskou použitelnost, prostředí, integraci a testování

samotnými uživateli. Jestli tedy něco opravdu chybí, pak to je holistický (celistvý) pohled na měření kvality kódu.

Zlepšení by se měla dočkat také správa testů. Jejich psaní je relativně přímočaré a pro jeho zjednodušení existuje spousta nástrojů. Takové nástroje ale již nenalezneme pro samotný management testů, které by umožňovaly spouštění správných testů během správných okamžiků v průběhu vývoje.

Samozřejmé je, že i zde se často setkáme s obyčejným nepochopením metodiky a jejím špatným používáním. Lze zmínit například kompletní ignorování testů a používání metodiky jen k popisu chování systému nebo aplikace. Stejně tak může docházet k abstrahování od psaní dodatečných testů, jelikož ne vždy přinesou viditelný užitek, nebo vývojový tým rovnou neví, jak s nimi dále pracovat a vyhodnocovat je. Tyto chyby jsou časté u týmů, které metodiku implementují jen z donucení, nebo na základě rady bez širšího kontextu.

Bylo by ovšem mylné přisuzovat tyto nedostatky přímo BDD, které za ně nemůže, ačkoliv se ho mohou přímo dotýkat. (Maayan, 2017)

# 4. Závěr

Ačkoliv na první pohled nemusí být spojení mezi obsedantně kompulzivními poruchami a metodikou TDD úplně zřejmé, poznatky z práce jasně ukazují, že existuje. Jeho závažnost a dopady nejsou vždy úplně jasné a konkrétní, nicméně je nutné o nich dále uvažovat a neopomíjet je. Mohou sloužit jako jeden z mnoha argumentů pro přesun od TDD k BDD, nebo jiné vylepšené metodice, popřípadě pro vývoj metodiky úplně nové.

Představené řešení problému v podobě BDD lze považovat za zajímavý směr, kudy by se metodiky vývoje softwaru mohly v budoucnosti dále ubírat. I přesto bylo naznačeno, že vývoj řízený chování taktéž není dokonalý a má další prostor k rozvoji.

Hlavní myšlenka a závěr práce by proto mohly být shrnuty následovně. Namísto hledání konečného řešení a jediné dokonalé metodiky by bylo vhodné se zaměřit na neustálý vývoj těch stávajících, a to nejen z pohledu programátorského, ale i byznysového a lidského.

# 5. Zdroje

**MAAYAN, G. David.** (2017). *Is TDD a Form of OCD?*. [online] Dostupné z: https://www.infoq.com/articles/tdd-ocd

**Informed Health Online**. (2017). *Obsessive-compulsive disorder: Overview*. [online] Dostupné z: https://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0072746/

**Agiledata.org**. (2017). *Introduction to Test Driven Development (TDD)*. [online] Dostupné z: http://agiledata.org/essays/tdd.html

**SCHOOTS, Huib**. (2014). *Test-Driven Development: Good or Bad?*. [online] Agile Record. Dostupné z: http://www.agilerecord.com/test-driven-development-good-bad

**NORTH, Dan**. (2006). *Introducing BDD*. [online] Dostupné z: https://dannorth.net/introducing-bdd/

**Jbehave.org**. (2016). *BDD, Getting Started.* [online] Dostupné z: http://jbehave.org/reference/latest/getting-started.html