Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Studijní obor: Informační systémy a technologie

Název semestrální práce:

A Quick guide to implementing ATDD

Autoři: Bc. Barbora Mlejnková

Bc. Jurij Povoroznyk

Předmět: 4IT421 Zlepšování procesů budování IS

Semestr: ZS 2018/2019

Abstrakt

Cílem této semestrální práce je shrnout principy přístupu Acceptance Test-Driven Development a představení jeho implementace. Tohoto cíle bychom chtěli dosáhnout vysvětlením základních principů tohoto přístupu a detailním popisem jeho cyklu. Zaměříme se také na samotné akceptační testy a uvedeme praktický příklad. V poslední části práce pak bude popsán samotný postup pro implementaci tohoto přístupu. V závěru je pak shrnuta celá semestrální práce a zhodnocení cílů.

Klíčová slova

Acceptance test-driven Development, ATDD, cyklus, implementace.

Obsah

Úvod

V současné době existuje nespočet metod vývoje softwaru. Velmi rozšířenou metodikou je agilní vývoj, který je založen na iterativním a inkrementálním vývoji softwaru. Příkladem této metodiky může být Extrémní programování, Scrum, Lean Development a spoustu dalších. V této semestrální práci se zaměříme na popis metodiky pojmenovanou Acceptance Test-Driven Development (ATDD), která se soustředí především na psaní testů před samotným vývojem daného softwaru. Tento přístup poskytuje celému týmu jasnější představu o tom, jak bude fungovat konečný produkt a umožní každému zůstat zaměřen spíše na dlouhodobé cíle než na konkrétní řádky kódu, ve kterých není zcela jasné, jaký je konkrétní účel a cíl daného softwaru.

ATDD je velmi úzce spjat s Test Driven Development (TDD). Avšak ATDD zahrnuje mimo jiné akceptační testování v procesu a klade se mnohem větší důraz na týmovou spolupráci.

Vybrat správnou metodu vývoje softwaru je první krok k úspěšnému dokončení projektu. V dnešní době je to ještě větší oříšek, proto se vám pokusíme představit konkrétní metodu ATDD, jenž by mohla právě pro vás být tou pravou.

Cíl práce

Cílem této semestrální práce je shrnout principy přístupu Acceptance Test-Driven Development a představení jeho implementace.

Struktura práce

Semestrální práce je rozdělena na dvě části. V první části je popsána čtenáři metodika ATDD a ukázán rozdíl mezi velmi blízkou metodikou TDD. V závěru této části je znázorněn samotný cyklus této agilní metodiky. Druhá část je zaměřena postup implementace této metodiky.

1 Acceptance Test Driven Development

Acceptance Test Driven Development (ATDD) přístup zahrnuje nejen členy vývojářského týmu, ale i ostatní zainteresované strany (obchodní partneří, zákazník apod.), kteří představují různé pohledy na daný software. Tito členové spolupracují při psaní konkrétních akceptačních testů před samotnou implementací odpovídající funkce či softwaru. Rozhovory, které se často vyskytují při generování těchto testů, jsou označovány jako „tři amigos“, jenž představují tři perspektivy. První perspektivou je konkrétní zákazník, který nejčastěji pokládá otázku „Jaký problém se snažíme vyřešit“. Další perspektivou je vývoj, který se snaží nalézt odpověď „Jak bychom tento problém mohli vyřešit?“ a poslední je testování . Zde si nejčastěji pokládají otázku „A co když...“. [1] Tato technika slouží především k přivedení zákazníka do procesu návrhu testů před zahájením samotného vývoje.

Akceptační testy jsou z pohledu uživatele, představují tedy externí pohled na systém. Nejčastěji se zaměřují na viditelný výstup systému po zadání konkrétního vstupu uživatele. Mimo jiné se pomocí těchto testů mohou kontrolovat další náležitosti softwaru, pokud se například jedná o e-commerce, tak se zde pomocí těchto testů kontroluje změna objednávky („přijata“ na „zaplacena“). V současné rozvíjející se době je další důležitou kontrolou samotné testování kooperace různých rozhraní mimo vyvíjený systém, např. sdílené databáze nebo webové služby.

Neúspěšné testy poskytují rychlou zpětnou vazbu, která sděluje, že konkrétní požadavky nejsou naplňovány. Tato zpětná vazba umožní vývojářům ihned reagovat a napravit konkrétní chyby bez nutnosti nechávat testovat konkrétní funkcionalitu stále dokola.

ATDD zahrnuje mnoho podobných postupů jako Story Test Driven Development (SDD) nebo Behavior Driven Development (BDD). Nejvíce ale ATDD souvisí s metodikou Test Driven Development (TDD), jak bude vysvětleno v následujících kapitolách. Přestože všechny tyto metodiky skrývají některé rozdíly, v celku vedou k podobným výsledkům jako právě ATDD.

1.1 Výhody

První a zásadní výhodou především pro zákazníka je celková doba dodání projektu, která je v průměru o 30-40% rychlejší. Dalším plusem využití této metodiky je výskyt samotných defektů, kdy se díky využívání ATDD vyskytuje pouze čtvrtina defektu ve srovnání s

klasickým přístupem vývoje (testovým přístupem). Snižuje budoucí náklady, které jsou nutné pro nápravu konkrétních defektů, jenž se vyskytnou příliš pozdě, a tudíž bude nutné pozměnit některé části již vyvinuté aplikace. [2]

Hlavní výhodou, která již byla zmíněna výše je úzká kooperace mezi jednotlivými členy. Díky této spolupráci jsou jasně definovány specifikace podle příkladu a také určení cíle, jenž se od konkrétního vyvíjeného softwaru očekává. Tato metodika zároveň poskytuje managementu přehled o tom, v jaké části vývoje se v současné době nacházíme, a to především díky % prošlým testům.

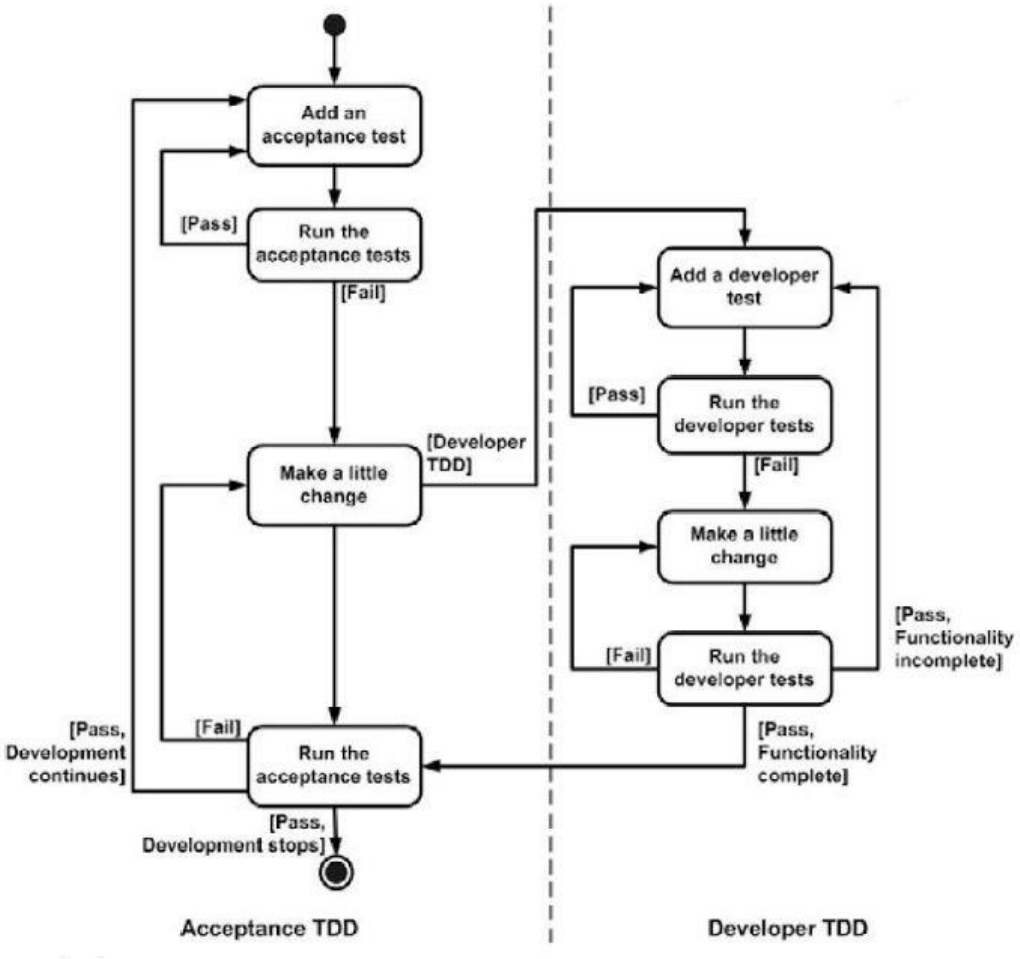
1.2 Nevýhody

Tato praktika je mimo automatizované testování spojená s využíváním specifických nástrojů, jako je například FIT/FitNess, Cocumber nebo jiné. A proto jedním z největších rizik této metodiky je právě samotná komunikace mezi jednotlivými členy týmu a zákazníkem, která může brzdit vývoj softwaru využíváním těchto nástrojů. Zvyšuje celkový čas vývoje a vyžaduje odbornou přípravu, mentoring a podporu.

Celkově musí tato metoda úzce spolupracovat se zákazníkem a konkrétními uživateli, kteří jsou nutní k úspěšnému dokončení projektu. Ovšem ne vždy mají čas nutný k upřesnění vyvíjeného softwaru.

2 ATDD vs TDD

Jak již bylo zmíněno, ATDD se velmi podobá metodice TDD, jak napovídá i její název. Test-Driven Development (TDD) se však zaměřuje pouze na perspektivu "zevnitř", což znamená, že vytváříme testy z pohledu vývojáře: "Je tento kód správný?" Toto je problém, který stojí za vývojem řízeným testováním. Tato metodika se především zaměřuje na jednotkové testy. Vývojář převezme požadavek a poté jej převede do konkrétního zkušebního případu. Poté zapíše kód, kterým předá pouze konkrétní testovací případy. Cílem této praxe je zabránit zbytečným aktualizacím, které nevyhovují požadavkům. [4] Tato metodika nutí vývojáře zaměřit se na konkrétní požadavky zákazníka a otestovat, zda jsou tyto požadavky následně splněny.



Obrázek 1 *Rozdíl ATDD vs TDD (Zdroj:[3])*

Vývoj založený na testování (TDD) se skládá ze čtyř kroků, jak je znázorněno na Obrázek 1. Prvním krokem je přijmutí požadavku a přetvoření na konkrétní test. Dalším krokem je spuštění těchto testů, které byly již vytvořeny. Tyto testy pochpitelně selžou, protože stále nebyl napsán žádný kód. Psaní kódu tedy následuje až po definování jednotkových testů, a to formou rychlých přírůstků. Programátor se tedy pokouší o to, aby daný test prošel. Pokud test neprojde, programátor změní kód. To se opakuje, dokud není test splněn.

Téměř stejné kroky má i samotný ATDD, kde se ovšem nespouští vývojářské testy, ale akceptační testy. Pokud se nepodaří nějaký test dokončit, vrací se opět ke změně k vývojáři, který opraví daný defekt. V opačném případě se úspěšně dokončí vývoj a aplikace je připravená k předání zákazníkovi.

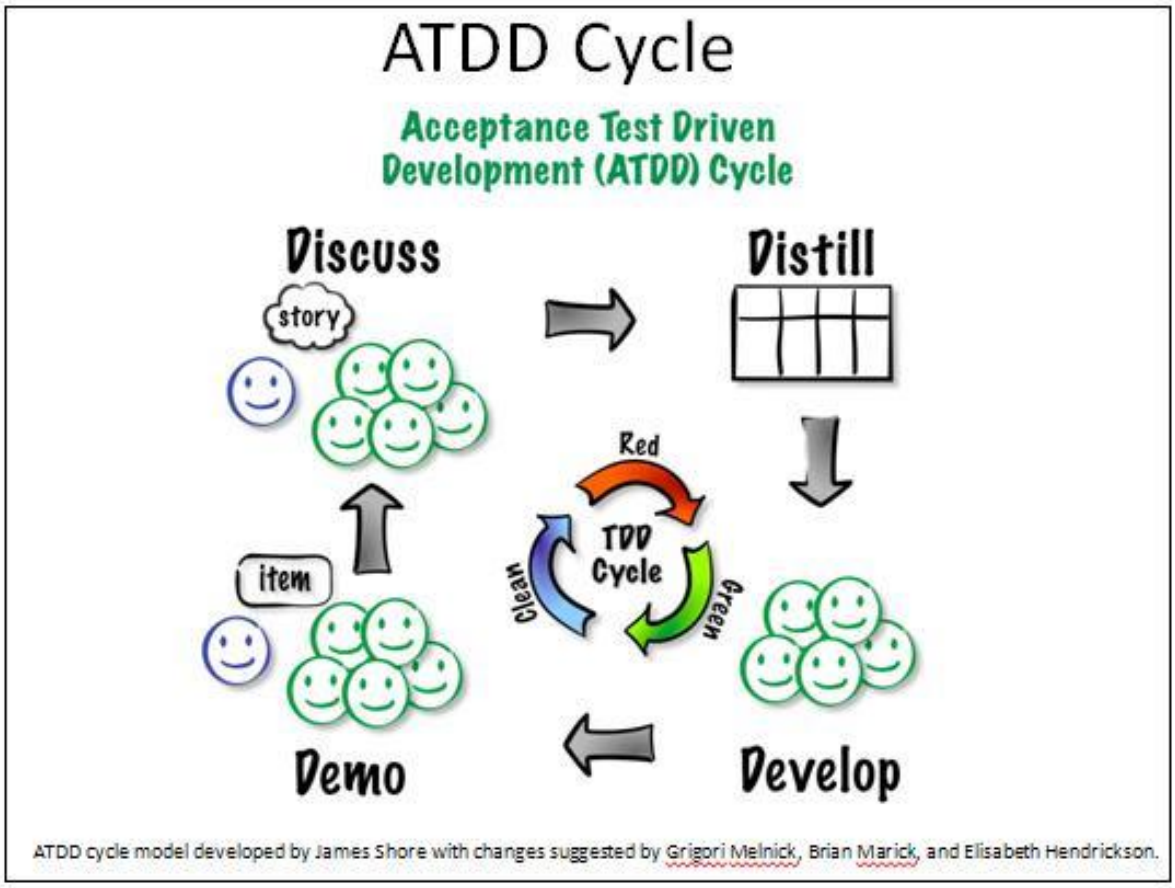
Z výše uvedeného obrázku je patrné, že se tyto metodiky doplňují. Nemůžeme tedy říci, že se zásadně liší. Ovšem mají některé rozdíly, mezi které patří:

- **Vztah** – ATDD se zaměřuje na vztah zákazníka, vývojáře a testera. Oproti tomu TDD se orientuje pouze na vývojáře, kterému pomáhá dodržovat konkrétní požadavky zákazníka, díky předem vytvořeným testům.
- **Zaměření** – TDD je zaměřeno pouze na tvorbu tzv. jednotkových testů, které ověřují funkčnost napsaného kódu. Oproti tomu ATDD je zaměřeno na akceptační testy, které jsou vytvářeny s konkrétním zákazníkem pro kterého je daný software či systém tvořen.
- **Obtížnost** – Jednotkové testy jsou velmi jednoduché na správu a lze s nimi dobře manipulovat a případně upravit. Oproti tomu ATDD je velmi složitá metodika na správu a samotný dohled, jak je již popsáno v kapitole 1.2. Souvisí s tím i konkrétní nástroje, které jsou zapotřebí pro správné fungování ATDD.

Hlavním rozdílem těchto dvou přístupů je především úroveň, na které se s nimi pracuje. Zatímco u TDD se pracuje na nízké úrovni s jednotkovými testy, ATDD pracuje na vysoké úrovni s akceptačními testy, kde je zapotřebí úzká komunikace se zákazníkem. Zároveň je TDD spíše více technické povahy oproti ATDD, kde jde hlavně o pochopení daného cíle a přenesení určité funkcionality do vyvíjené aplikace.

3 Cyklus ATDD

Jak již bylo zmíněno výše ATDD rozšiřuje a doplňuje klasický vývoj TDD. Vývoj řízený akceptačními testy se rozšiřuje především o typické jednotkové testy, které se snaží otestovat, zda se systém chová dle očekávání uživatelů (zákazníka). ATDD celkově pak prochází jednoduchým cyklem, který je zobrazen na Obrázek 2. Cyklus se skládá ze čtyř části, které jsou popsány níže.



Obrázek 2 *ATDD Cycle (Zdroj:[5])*

3.1 Discuss

První část je zaměřená na konzultaci požadavků s konkrétním zákazníkem. Do této diskuze je zapojen celý agilní tým a konkrétní obchodní partneři. Zde se určí požadované náležitosti na systém, funkčnost a další důležité detaily nutné k úspěšnému přijetí daného softwaru. Zároveň se pokládáním otázek členů týmu objasní chování systému. Po této diskuzi by měly členové týmu pochopit konkrétní cíl vyvíjené aplikace. Na konci diskuze agilní tým sepíše

přijímací testy. V okamžiku, kdy jsou tyto testy úspěšné, lze považovat vyvíjený produkt za hotový.

3.2 Distill

Další fází je převedení specifikovaných testů do konkrétního testovacího frameworku. Existuje celá řada automatizačních rámců. Nejznámější z nich jsou FIT, Fitnesse, Concordian nebo Robot Framework. Samotné testy jsou psané ve formátu Given When Then[1]. [6] V těchto testech by měly být napsané veškeré požadavky zákazníka, které byly zmíněné v předešlé fázi tohoto cyklu Discuss. Další testy se mohou přidat v pozdějším vývoji, kdy se zcela pochopí smysl tohoto systému. Výsledkem této fáze je seznam testů, jenž pokrývá požadavky zákazníka.

3.3 Develop

Předposlední fází tohoto cyklu je samotný vývoj aplikace. Vývojáři se zde řídí metodikou TDD. Pokud se podaří dokončit všechny testy, vývojář ručně ověří integraci jednotlivých částí, tedy zda vše správně funguje. Také v této fázi může nastat situace, která přinutí konkrétního vývojáře přidat některé testy, na které se zapomnělo v předchozí fázi. Tyto nově přidané testy musí však přidat taktéž do předchozího souboru, aby bylo možné sdílet je se všemi zúčastněnými aktéry. Výsledkem této části je samotný software a případně nově přidané komplexní testy.

3.4 Demo

Tato část je zaměřena především na předání konkrétní demo aplikace zákazníkovi. Předem stanovené akceptační testy jsou použity k ověření všech požadovaných funkcí. Vývojáři si musí být zcela jistí úspěšným dokončením veškerých testů sepsaných v první fázi tohoto cyklu. Úspěch či neúspěch jednotlivých testů bude ukázán přímo v demo aplikaci. Po dokončení dema zákazník schválí aplikaci a tím je konkrétní software a samotný projekt úspěšně dokončen. V opačném případě se upřesní detaily a doladí nesrovnalosti na vyvíjené aplikaci. [6]

[1] GWT je polo strukturovaný způsob zápisu testovacích případů, viz. kapitola 5

4 Akceptační testy

Akceptační testy jsou testy, které signalizují, zda lze daný projekt či jeho část povaovat za splněnou. Nejlépe lze akceptační test popsat konkrétním příkladem:

Mějme manažera marketingu, který požaduje jako jednu z funkcionalit poskytnutí slevy věrným zákazníkům. Níže jsou sepsány detaily tohoto user story:

Pokud je hodnocení zákazníka Dobré a celková cena objednávky je nižší nebo rovna 100 Kč,

pak systém neposkytne slevu,

jinak systém poskytne slevu 1%.

Pokud je hodnocení zákazníka Výborné,

pak systém poskytne slevu 1% na jakoukoli objednávku.

Pokud je celková cena objednávky větší než 500 Kč,

pak systém poskytne slevu 5%.

Pokud položíme jednoduchou otázku – Jakou slevu systém poskytne pro zákazníka, který má hodnocení Dobré a jehož objednávka je v celkové výši 500,05 Kč?

Odpovědí se zde nabízí, i přes na první pohled vyčerpávající popis požadavku, více. Záleží, jak se k danému pravidlu postavíme, řešením může být 1, 5 nebo dokonce 6%. Pravidlo tedy není jednotné. [7]

Řešením může být například následující tabulka 1. Zápis však může být proveden i zmiňovaným GWT formátem.

Celková cena	Hodnocení	Sleva
100 Kč	Dobré	0%
100, 01 Kč	Dobré	1%
500,01 Kč	Dobré	1%
0,01 Kč	Výborné	1%
500,00 Kč	Výborné	1%
500, 01 Kč	Výborné	5%

Tabulka 1: Tabulka formulací akceptačních kritérií (Zdroj:[7])

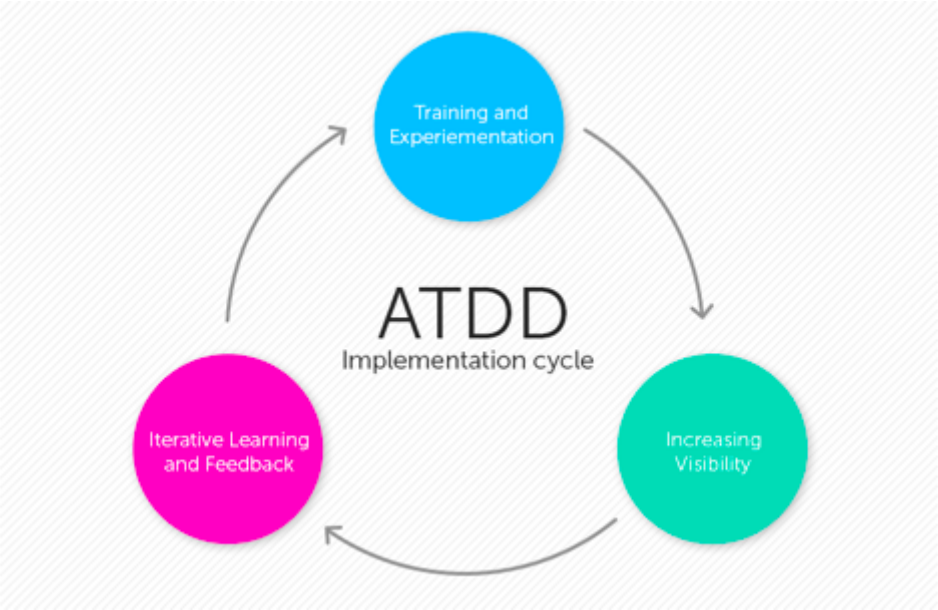
Vzájemnou spoluprací více členů týmu a tím i více pohledů na věc ihned ze začátku lze předcházet pozdějším nesrovnalostem.

5 Implementace ATDD

Raj Subramanian [8] popisuje ve svém článku A Qucik Guide to Implementing ATDD rozděluje proces implementace ATDD do 3 základních kroků:

- trénink a experimentování,
- zvyšování povědomí,
- iterativní učení a poskytování zpětné vazby;

Tyto 3 kroky jsou znázorněny na následujícím obrázku:



Obrázek 3: Cyklus implementace ATDD (Zdroj: [8])

5.1 Školení a experimentování

Pokud v týmu pracovníků přihlédneme k jejich individuálním pracovním zkušenostem - může se jednat jak o zkušenosti ze stejné firmy, avšak v jiném týmovém složení na jiném projektu, či o zkušenosti práce v týmu z organizace jiné - získáme mnoho způsobů jak v týmu přistupovat k řešení jednotlivých úkonů. Abychom předešli nejasnostem týkajících se cílů nejen týmu samotného, ale i organizace jako takové, je důležité poskytnout pracovníkům vhodné školení. V případě ATDD jde zejména o školení zahrnující představení cílů a očekávání doplněné o konkrétní informace o výsledcích, které přináší využívání tohoto přístupu. Základními přínosy pak bývají zvýšená týmová spolupráce, lepší pochopení

požadavků, dřívější identifikace defektů v rámci životního cyklu tvorby softwaru (SDLC, Software Development Life Cycle) nebo zapojení celého týmu do stanovení jasných akceptačních kritérií. Toto školení slouží nejen pro samotné týmové pracovníky, ale i ostatní zainteresované strany.

Po provedeném školení přichází na řadu částečná implementace přístupu. Pracovníci tak mají možnost vyzkoušet si základní principy a poskytnout iterativní zpětnou vazbu. Subramanian uvádí příklad takovéto zkušební implementace. V tomto příkladu rozděluje tým 25 lidí do dvou skupin - tým A a tým B. Tým A čítá 12 lidí a tým B se skládá ze zbylých 13 lidí. Pro tým A je pak připraven 2denní tréninkový workshop. Na tomto workshopu jsou týmu A představeny základní koncepty které jsou vhodné pro daný projekt, který tým řeší a také například jak v tomto procesu využít automatizace. Tým A tedy při dalším řešení prací na projektu aplikuje ATDD přístup. Tým B je pak tím týmem lidí, kteří pracují dosavadním způsobem. Tímto způsobem tedy lze snadněji demonstrovat představované přínosy implementace ATDD a jednotliví členové týmu pak mohou poskytnou zpětnou vazbu k tomuto druhu přístupu. [8]

Pro samotný workshop lze využít několik teoretických i praktických cvičení. Příkladem je Given/When/Then (GWT) cvičení. GWT formulace jsou jakousi šablonou pro definici akceptačních testů konkrétního user story. Skládá se, jak název napovídá ze 3 základních pojmů:

- Given – počáteční kontext,
- When – akce,
- Then – očekávaný výsledek;

Dalšími pojmy jsou pak například And nebo But. [9]

5.2 Zvyšování povědomí

Během jednotlivých sprintů, kterými projekt prohází je velice snadné ztratit přehled o jednotlivých procesech, které je nutné dodržovat. Klíčem k řešení této situace je zpřístupnění cílů a očekávání celému týmu.

Během implementace ATDD je tedy vhodné začít sepsáním všech základních procesů, které je nutné provést. Tento seznam pak Subramanian umísťuje nejlépe tam, kde tým pořádá

stand-up meetingy. Pro každý user story pak přidává check list jednotlivých položek, které je třeba splnit, aby mohl být daný úkol považován za hotový. Jednou takovou položkou je například kick-off meeting. Při tomto meetingu diskutují společně vývojář, tester a ostatní zainteresované strany o požadavcích jako jeden tým. Jsou diskutovány jednotlivé požadavky a možnosti jejich automatizace, konkrétní akceptační kritéria definována prostřednictvím GWT formulací. Další položkou jsou tzv. QA-Dev Handoff. Zde vývojář představuje testerovi prostřednictvím ukázky či diskuze, jak byly požadavky řešeny a jaké jednotkové testy se týkají řešeného požadavku. Takto tester snadněji pochopí implementovanou funkcionalitu a zjistí, které části nejsou pokryté jednotkovými testy. Celý tento proces pak zajišťuje efektivnější a větší pokrytí kódu testy. Přestože některé položky checklistu mohou být zřejmé, je na místě je definovat všechny. Příkladem je například potvrzení, že veškeré defekty, které souvisí s daným požadavkem byly vyřešeny apod.

Další metodou, kterou Subramanian ve svém návodu popisuje je tzv. „Process Hawk“. Jde o jednoho člena týmu – může jít o scrum mastera, projektového manažera či kohokoli kdo zajistí, že tým bude postupovat dle definovaného procesu. Dodržování ATDD procesů je pak tímto členem připomínáno ostatním členům týmu na jednotlivých schůzkách, stand-upech a retrospektivách. [8]

5.3 Iterativní učení a poskytování zpětné vazby

Neustále poskytování zpětné vazby je základem při implementaci jakéhokoliv agilního přístupu, ani ATDD není výjimkou. Pořádáním retrospektivních meetingu jednou za 7 či 14 dní pomůže identifikovat ty procesy, které přispívají k efektivitě týmové práce a ty, které je třeba upravit. Udržování procesů, které nepřináší žádnou hodnotu je pro každý podnik pouze ztrátou času, financí a energie. Zde stojí za zmínku metodologie Lean Startup, která poskytuje přístup ke vytváření a správ startupů a podporuje rychlejší získání požadovaného produktu. Metodologie v podstatě říká, kdy a jak vhodně začít a řídit tvorbu produktu, za jakých podmínek v tomto procesu pokračovat, nebo naopak kdy zastavit. Jedná se o principiální přístup k vývoji jakéhokoliv nového produktu. [10]

V návodu podle Subramaniana jsou navrženy 30-ti minutové meetingy jednou za týden. Tohoto meetingu se účastní vytvořený tým A. Jeho cílem je zjistit, jak úspěšně se týmu daří implementovat navržené procesy. Díky poskytnuté zpětné vazbě je pak mnohem jednodušší identifikovat neefektivní či zbytečné procesy. Mimo tyto týdenní meetingy pak Subramarian

pořádá ještě retrospektivní meetingy, které se uskutečňují po dvoutýdenních sprintech. Zde je pak možné přidat i zpětnou vazbu poskytnutou týmem B. Dobrým zdrojem informací mohou být samozřejmě také denní stand-upy. [8]

5.4 Výsledky

Výsledkem popsaného implementačního cyklu jsou rozdíly, které pociťují členové týmu A zejména v oblastech týmového ducha, spolupráce a definování požadavků.

Tým A většinou vnímá lepší práci v týmu, kdy se každý člen podílí na daném user story od jeho začátku až do konce. Zajišťuje tedy, aby byly poskytnuty veškeré potřebné informace pro vývoj i otestování. Má také přehled o navazujících činnostech a na konci sprintu je také zodpovědný za demonstrování vyřešeného user story zainteresovaným stranám. Tento pocit vlastnictví každého člena pak výrazně posiluje celý tým.

Co se týká spolupráce, důležitou roli zde sehrávají zmiňované kick-off meetingy, které spojují obchodní partnery, vývojáře a testery. Dohromady tak mohou lépe porozumět jednotlivým požadavkům. QA-Dev Handoff pomáhá vývojáři i testerovi rozpoznat, které části funkcionality testovat a zároveň opět podporuje lepší pochopení požadavku. Jednotlivé zainteresované strany tak téměř neustále komunikují mezi sebou navzájem.

V neposlední řadě jsou výsledky vidět i na požadavcích na software. Jedním z hlavních problémů dosavadních procesů bývá velice častá změna požadavků. S použitím ATDD jsou veškeré požadavky neměnné, jakmile vývojář začne s prací. Jakékoliv další změny pak musí být založeny na nových user stories zařazených do nadcházejících sprintů. Vzhledem k velkému důrazu na spolupráci všech stran na začátku procesu, a hlavně na definici akceptačních testů jsou však takovéto změny minimalizovány. [8]

Jakmile tedy samotný tým B vidí jasné rozdíly, implementuje ATDD stejně jako tým A. Výsledkem je celý původní tým pracující pomocí metodiky ATDD.

Závěr

ATDD je metodika, která se snaží vést k testování a vývoji co nejdříve na začátku životního cyklu vývoje softwaru. Přináší nám lepší přehlednost, lepší týmovou spolupráci a podporuje jasné pochopení požadavků všemi stranami již v začátku vývoje.

Přesto však není ATDD vhodným řešením pro všechny projekty. Existuje mnoho dalších agilních metodik, které mohou danému projektu vyhovovat více. Nicméně v projektech, ve kterých je kladen důraz na jasnost akceptačních kritérií a posílení spolupráce jednotlivých zainteresovaných stran a členů týmu, tato metodika přináší velkou podporu.

Seznam literatury

[1] Agile Alliance. *Acceptance Test Driven Development (ATDD)* [online]. 16. 2. 2016 [cit. 25. 11. 2018]. Dostupné z: https://www.agilealliance.org/glossary/atdd/

[2] cPrime. *An Introduction to Acceptance Test-Driven Development* [online]. 5. 1. 2015 [cit. 25. 11. 2018]. Dostupné z: https://www.slideshare.net/cPrime/an-introduction-to-acceptance-testdriven-development

[3] SELECKÝ, Matúš. Arduino 2016. ISBN 9788025148495, Computer Press, Albatros Media a.s.

[4] AMBLE, Vinai. *Model View Presenter (MVP) in Android, Part 1* [online]. 31. 5. 2018 [cit. 27. 11. 2018]. Dostupné z: https://www.simplilearn.com/agile-acceptance-test-driven-development-article

[5] CHANDANA. *Agile Acceptance Test Driven Development: Agile Certification Training* [online]. 7. 4. 2017 [cit. 26. 11. 2018]. Dostupné z: https://www.ca.com/en/blog-agile-requirements-designer/guide-to-test-driven-development-tdd-vs-bdd-vs-atdd.html

[6] EVANGELISTI, Augusto. *ATDD* [online]. 15. 4. 2015 [cit. 26. 11. 2018]. Dostupné z: https://mysoftwarequality.wordpress.com/tag/atdd/

[7] PUGH, Ken, 2011. *Introduction to Acceptance Test-Driven Development. A Net Objectives Essential White Paper* [online]. NET OBJECTIVES, INC. [cit. 30.11.2018]. Dostupné z: https://www.netobjectives.com/files/resources/articles/Introduction_AcceptanceTestDrivenDe velopment.pdf

[8] SUBRAMANIAN, Raj, 2018. *A Quick Guide to Implementing ATDD. InfoQ* [online]. [cit. 12.10.2018]. Dostupné z: https://www.infoq.com/articles/quick-guide-atdd

[9] CUCUMBER, 2018. Gherkin Reference. GitHub [online]. [cit. 29.10.2018]. Dostupné z: https://docs.cucumber.io/gherkin/reference/

[10] BLANK, Steve, 2013. Why the Lean Start-up Changes Everything. *Harvard Business Review* [online]. [cit. 30.10.2018]. Dostupné z: https://hbr.org/2013/05/why-the-lean-start-up-changes-everything