Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	LS 2016/2017
Autoři – jméno, příjmení, xname	Vojtěch Švanda, xsvav14; Dominik Firla, xfird00
Téma	**The MINDSET of the AGILE DEVELOPER**
Datum odevzdání	28. 4. 2017

Abstrakt:

Tato semestrální práce se zabývá agilním způsobem myšlení. Nejprve popisuje hlavní hodnoty agilního vývoje a principy agilního myšlení. Následně vysvětluje, jak lze agilní myšlení rozvíjet. Nakonec na některých běžných agilních metodikách zmiňuje časté chyby způsobené neagilním přemýšlením při implementaci agilních metodik.

Klíčová slova:

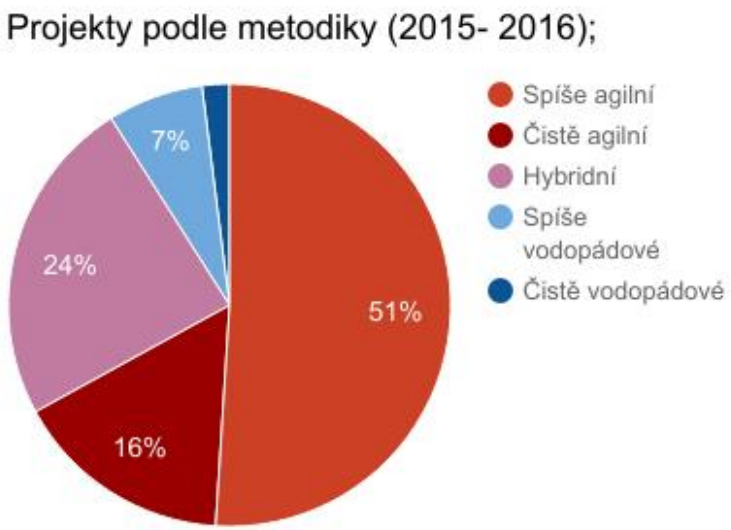
agilní způsob myšlení, agilní vývoj, tým, proces, metodiky, rozvoj

Obsah

1 Úvod

Trendem ve vývoji softwaru je v posledních letech přechod od tradičních vodopádových metodik k agilním metodikám vývoje. Tento trend ilustruje obrázek 1, ve kterém jsou rozděleny projekty podle metodik. Tato transformace s sebou nese velké změny především v aplikaci úplně jiných principů, než byly doposud používané, a také v potřebě změnit způsob přemýšlení jedinců, kteří tyto principy používají (Jeremiah, 2016).

Hlavním cílem této práce je poukázat na rozdíly mezi agilním a tradičním myšlením při vývoji softwaru a také přiblížit některé způsoby rozvoje agilního způsobu myšlení.



Obrázek 1 Rozdělení používaných metodik na projektech, *(Jeremiah, 2016)*

Pro definování agilního způsobu myšlení je použita práce psycholožky Carol Dweckové, která způsoby myšlení rozděluje do dvou hlavních skupin – růstové a fixní myšlení (Popova, 2015). Specifika a problematika transformace člověka z fixního myšlení na růstové je blíže vysvětlena v rámci kapitoly 2.1.

V poslední části práce jsou popsány některé běžné agilní metodiky. U každé z nich jsou stručně představeny běžné chyby, kterých se týmy při jejich implementaci dopouštějí. Některé z těchto chyb vyplývají právě z neagilního způsobu myšlení.

2 Agilní způsob myšlení

Hlavní myšlenky dnešního agilního vývoje vycházejí z tzv. Manifestu Agilního vývoje softwaru, který vznikl v roce 2001. Manifest definuje základní hodnoty a celkově 12 principů agilního vývoje softwaru, které se například zaměřují na jednotlivce a jeho schopnosti či komunikaci se zákazníky, přičemž jsou vítány změny i v pozdějších fázích vývoje softwaru. Základními hodnotami agilních metodik jsou pak následující poučky, kdy má přednost:

***Jednotlivec** a **interakce** před procesy a nástroji*
***Fungující software** před vyčerpávající dokumentací*
***Spolupráce se zákazníkem** před vyjednáváním o smlouvě*
***Reagování na změny** před dodržováním plánu*

Ačkoliv body na pravé straně jsou chtěné, často i potřebné, tak přesto jsou více ceněné body nalevo, které mají pro firmu i pro zákazníka mnohem větší hodnotu a zvyšují jeho konkurenceschopnost (Beck, a další, 2001).

Tento přístup byl nejdříve spíše okrajovou záležitostí, ale s odstupem času lze říci, že v současné době dochází ke stále větším a rychlejším změnám v oblasti technologií a vývoje softwaru, a proto je agilní vývoj softwaru nejen oblíbenější ale také potřebnější, a tak firmy opouští tradiční vodopádové modely a věnují nemalé úsilí do restrukturalizace a transformace na agilní metodiky. To však nemusí vyhovovat vývojářům a dalším členům týmu, kteří často nemají na výběr a musejí přejít právě z klasického vodopádového modelu na agilní metodiku. V takovém případě je nutností vysvětlit členům týmu, jakým způsobem agilní vývoj chápat a na jakých principech stojí. Jinak se může stát, že členové týmu budou pracovat v agilní metodice, ale přesto budou nad prací přemýšlet a uvažovat tradičním způsobem, který je vázaný na vodopádové modely (Kolektiv autorů, 2016) (Broza, 2016) (Mysliveček, 2011).

2.1 Principy a způsoby myšlení v agilním vývoji

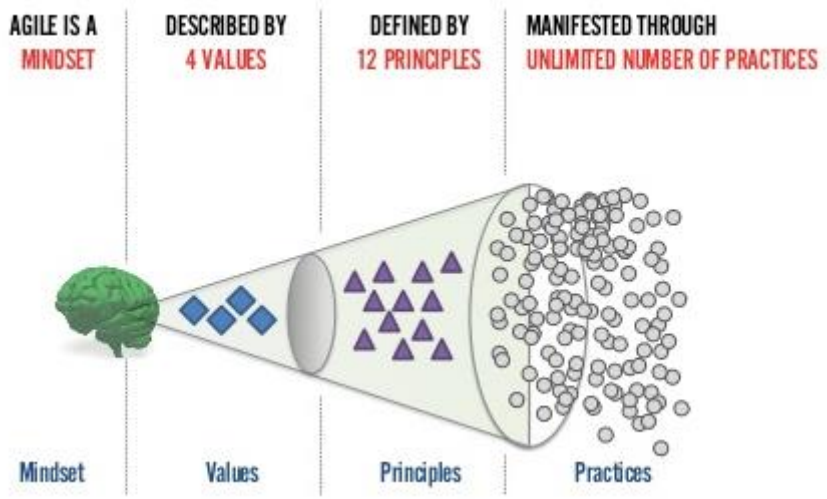
V dnešní době se často stává, že agilní metodiky jsou brány jako sada postupů či proces, kterým je nutné se řídit. Díky tomu lze popsat transformaci či přechod na agilní přístupy pomocí pár kroků, které zachycují podstatu agilního přístupu (Broza, 2016) (Denning, 2016):

- Nejdříve dojde k sestavení malého multifunkčního týmu.
- Člověk z byznysu je dosazen do role vlastníka produktu a bývalý projektový manažer má za úkol řídit nově implementovanou metodiku.
- Úkoly jsou psány na lepící papírky nebo se udržují pomocí nějakého nástroje.
- Plánuje se pomocí hodnocení pracnosti daného úkolu.

- Usiluje se o průběžnou integraci, automatizaci testování a pravidelné zlepšování kódu.
- Vytvořená práce se předvádí zákazníkům jednou za několik týdnů (nejčastěji mezi 2-4 týdny).

Agilní přístupy by ale neměly být brány jako dané procesy, které je nutno přesně dodržovat, ale jako určitý druh myšlení, které lze použít k vyřešení daného problému v dané problematice, a to právě pomocí metodiky využívající myšlenky agilního vývoje – samotné procesy nejsou důležité, ale proces zlepšování procesů ano. Proto je také nezbytné, aby firma, která chce využívat výhod agilních metodik, naučila své zaměstnance agilní přístupy i s agilním způsobem myšlení. V současné době je agilní způsob myšlení jedním z nejdůležitějších součástí agilních metodik, a to i přesto, že není zmíněný v Manifestu agilního vývoje softwaru (JustSolve, 2016) (Denning, 2016) (Broza, 2016).

Na obrázku 1 je vidět důležitost agilního způsobu myšlení. Tento způsob myšlení je pak vysvětlován čtyřmi hodnotami, které jsou definovány již zmíněnými 12 principy. To vše je spojeno v praktikách, kterých může být neomezené množství. Mezi tyto praktiky patří například SCRUM, Kanban či extrémní programování. Také mezi ně mohou patřit vlastní agilní praktiky a procesy. Z obrázku 1 lze vyčíst také jasný rozdíl mezi pojmy „být agilní“ a „používat agilní postupy“. Když je tým agilní, tak vychází z levé strany, tedy od způsobu myšlení. Pokud tým „jen“ používá agilní metodiku, tak vychází z pravé strany od praktik a dosažení agilního způsobu myšlení je těžší (Sidky, 2015) (Booz Allen, 2016).



Obrázek 2 Agilní způsob myšlení až po agilní praktiky, *(Sidky, 2015)*

K definici agilního způsobu myšlení lze použít rozdělení podle psycholožky Carol Dweckové. Ta určila dva základní druhy způsobů myšlení – růstové a fixní. Fixní myšlení

se vyznačuje tím, že osobnost člověka, inteligence a kreativita jsou dané od přírody. Člověk s fixním myšlením si myslí, že nic z toho nelze nějakým smysluplným způsobem změnit a že úspěch je výsledkem inteligence dané od přírody. Proto takový člověk usiluje jen o úspěch a za každou cenu se snaží vyhnout chybám a selhání. Naopak růstové myšlení se vyznačuje vědomím, že osobnost, inteligence a kreativita může být stále rozvíjena, a tak je opravdový potenciál člověka neznámý. Takový člověk má potěšení z výzev a také vidí chyby a selhání jako možnost se z nich poučit, spíš než jako nedostatek inteligence u fixního myšlení. Proto také lze za agilní způsob myšlení považovat právě růstové myšlení. Agilní způsob myšlení se dále vyznačuje těmito vlastnostmi (Popova, 2015) (Agile-Scrum, 2016) (Sidky, 2015) (Howard, 2015):

- Týmy vítají různé pohledy na danou problematiku a jsou za různé názory rádi. Je dobře když spolu v týmu pracují vývojáři, analytici a testeři. Díky tomuto spojení je zajištěn pohled z různých perspektiv.
- Členy týmu práce baví a jsou motivovaní a odhodlaní na daném produktu pracovat. Také to znamená, že tým nefunguje jen v práci a v rámci dané metodiky, ale také po práci v rámci teambuildingů a dalších akcí.
- Pracovní tempo je trvale udržitelné, což znamená, že jednotliví členové týmu mají stanovené úkoly a cíle podle svých schopností a znalostí.
- Členové týmu informují ostatní o své práci, a i o svých chybách nebo problémech.
- Lidé v týmu chtějí a musí spolu komunikovat a spolupracovat na řešených problémech.
- Členové týmu se stále učí a zlepšují nejen ve svém oboru, ale snaží se získávat znalosti a zkušenosti i od ostatních členů týmu.
- Znalosti jsou sdíleny dobrovolně a volně mezi členy týmu.

Oba druhy způsobů myšlení zakořeňují v člověku již od dětství, proto může být přechod na agilní způsob myšlení velmi těžký (Popova, 2015) (Sidky, 2015).

2.2 Rozvoj agilního způsobu myšlení

Rozvoj specifického myšlení chce více než jen využívat specifické nástroje, procesy a šablony. Základním kamenem je ochota učit se novým věcem a také otevřenost novým nápadům a myšlenkám ostatních členů týmu. Dále se k rozvoji nového myšlení velmi hodí vyprávění příběhů. Díky příběhům, které lidé uslyší a nějakým způsobem je prožijí, bude využita představivost, která pomůže k získání nového myšlení. Dále je důležitý způsob, jakým

mluvíme. Je nutné využívat slov k popsaní věcí tak, aby bylo možné si je jednoduše vizualizovat. Vizualizace pomáhá s formováním specifického myšlení. Pak je také samozřejmostí poučení se z chyb, a nejen z těch „náhodných“, ale také je nutné nebát se používat metodu pokus-omyl. Právě tato metoda přispívá k rozvoji agilního způsobu myšlení. Rozvíjet agilní způsob myšlení lze také následujícími způsoby (Denning, 2016) (Easuwaran, 2017) (Subramaniam, a další, 2006):

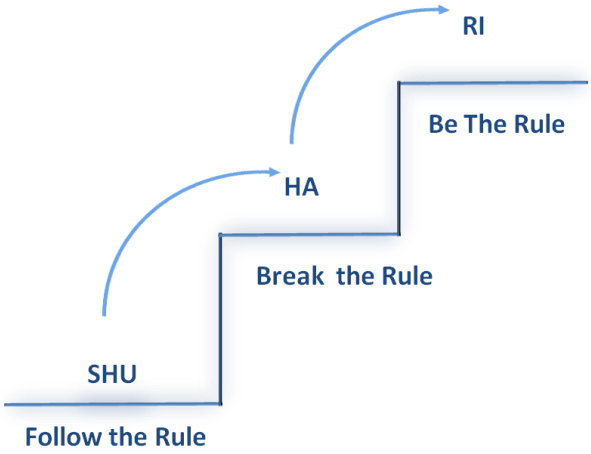
- Vedoucí (např.: manažer, projektový manažer atd.) by měli pracovat s týmem místo toho, aby tým pracoval pro ně. Tato změna vyžaduje hodně učení, přemýšlení a překonávání některých návyků.
- Vedoucí by měli nechat svému týmu určitou možnost se rozhodovat, jak řešit problémy, které nastaly nebo nastanou. Tým by měl mít možnost zjistit, jak správně problémy vyřešit.
- Je dobré začít s minimem informací, které jsou k dispozici a poté se postupně učit nové a potřebné věci. Není dobré chtít vědět vše do detailu, než vůbec začne práce na produktu.
- Společně s agilními praktikami vzniká určitý chaos. Tradiční přístupy by navrhovali ihned se chaosu zbavit, ale v agilních praktikách je chaos chtěný, ale pouze do doby, kdy je pod kontrolou a pomáhá týmu dosáhnout cílů a úkolů.
- Emaily by se měly používat jen pokud je potřeba něco zaznamenat. Jinak je dobré spoléhat na osobní setkání a komunikaci s danými lidmi. Tím dochází nejen k úspoře času, ale také k lepšímu pochopení věcí, které je potřeba vyřešit.

Celkově je důležité neupadnout do stereotypu a stále přemýšlet nad novými možnostmi, které agilní způsob myšlení umožňuje – agilitou by měla být především myšlena schopnost se přizpůsobit. Celkovým výsledkem pak může být způsob myšlení jako ve firmě Starbucks, ve které je barista na vrcholu a všichni ostatní ve firmě ho podporují. Důvod je jednoduchý, barista je tím, kdo vytváří firmě hlavní hodnotu. Barista nepracuje pro manažera, ale manažer se stará o baristu (Easuwaran, 2017) (Denning, 2016).

Příkladem takové metody, která se přímo hodí i používá k rozvíjení agilního způsobu myšlení, je Shu-Ha-Ri. Tato metoda pochází z japonských bojových umění (především z Aikida). Jelikož v agilních metodikách je nutné se často učit, tak je tato metoda, která popisuje cyklus učení, velmi výhodná a potřebná. Lidé jako Martin Fowler či Alistair Cockburn aplikovali tuto metodu na různá agilní prostředí. Přesto lze najít různé interpretace této metody,

což záleží hlavně na vlastních zkušenostech a oblasti zájmů jejích autorů. Sám Fowler doporučuje formulaci Clarka Terrryho (představitel jazzu) (Fowler, 2014) (Novack, 2016).

Metoda se skládá ze tří fází a jak napovídá samotný název, jednotlivé fáze jsou Shu, ha a ri. Každá z úrovní představuje určitou fázi v učení a znalostech člověka v dané oblasti. Jakmile dojde k ovládnutí jedné úrovně, tak lze přejít na vyšší. Tato poučka je ve svém principu velmi jednoduchá, ale často opomínána projektovými manažery i scrum mastery. Často se stává, že nezačínají první fází. Na obrázku 2 lze vidět postup mezi jednotlivými fázemi metody (Novack, 2016) (Šochová, 2013).



Obrázek 3 Jednotlivé stupně metody Shu-Ha-Ri, *(Dak, 2013)*

První fáze Shu je o učení se základů bez jakékoliv změny praktiky či metodiky. Důvodem tohoto přístupu je nutnost si zažít základy a naučit se používat veškeré postupy, tak aby byl naplněn význam agilních principů a myšlenek. To může trvat i 3 roky. Jde o vystavění kostry agilního způsobu myšlení. Vzhledem k původu z japonských bojových umění je člověk praktikující Shu studentem, tedy mu také nepřísluší, jakkoliv měnit či modifikovat praktiky stanovené a učené mistrem – nemá k tomu dostatek zkušeností (Šochová, 2013) (Novack, 2016).

Ve druhé fázi Ha má tým již základy zažité a vyzkoušené v praxi. Lze tedy začít s dalším rozvojem. Dochází k studiu různých doporučení, dalších praktik, teorií a metodik, ze kterých lze brát ponaučení a aplikovat je na situace a problémy, které se vyskytly v týmu. Vzhledem k původu v japonských bojových umění je člověk praktikující Ha pokročilým, tedy již může provádět menší inovace a vylepšovat základy, které se dokonale naučil – má potřebné znalosti ale stále ne dostatek zkušeností (Novack, 2016) (Fowler, 2014).

Třetí fáze Ri již naznačuje, že tým je již natolik vyspělý a zkušený, že nemá potřebu se přímo učit z jiných zdrojů a od jiných lidí. Tým se v této fázi učí hlavně ze svých vlastních

zkušeností a praxe. Takový tým je schopen vytvářen vlastní praktiky, doporučení i principy, na kterých mohou fungovat. Díky původu v japonských bojových umění, je člověk praktikující Ri již mistrem, tedy má dostatek zkušeností k úpravám současných praktik, ale i k vymýšlení nových, které může učit studenty ve fázi Shu (Novack, 2016) (Fowler, 2014).

Předtím než tým považuje svou situaci za výjimečnou, tak je nutné se podívat po nějakém běžném řešení. Také je důležité se ujistit, že tým velmi dobře chápe a zná základy toho, co je nutné udělat. Jednoduchým příkladem pak může být tým, kterému běžně trvaly denní standupy (viz kapitola 3.1 – Scrum) přes 35 minut a členové týmu dospěly k tomu, že je budou dělat méně často. Tuto situaci ale lze vyřešit pomocí metody Shu-Ha-Ri. Ve fázi Shu se tým při denním standupu vrátí ke klasickým třem otázkám, ve kterých je pokryto to, co dělal včera, co bude dělat dnes a s jakými problémy se setkal. Jakmile k tomuto došlo, tak se čas standupů výrazně zkrátil a také došlo ke zlepšení kvality informací. Ve fázi Ha pak díky dobrému pocitu ze zlepšení dochází k malým úpravám. Členové týmu si přidali čtvrtou otázku, ve které se ptali ostatních členů, zda nepotřebují s něčím pomoct. Celkově dochází ke zlepšení spolupráce a agilní způsob myšlení se stává součástí nejen standupů, ale i všech členů týmu. V poslední fázi Ri pak tým upustil od struktury otázek a jejich standupy spíše připomínali určitý tok informací důležitých pro tým a jednotlivé členy. To jim pomáhá s rychlou adaptací na případné problémy, které by se mohly při práci vyskytnout (Novack, 2016).

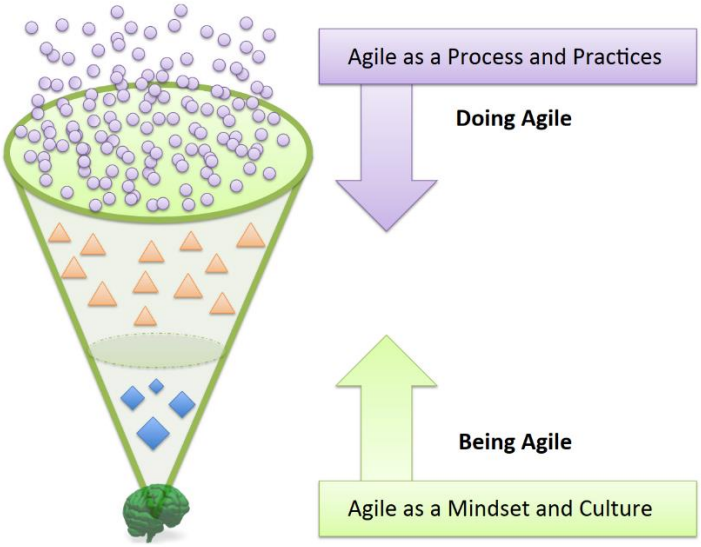
2.3 Rozdíly v pojmech být agilní a používat agilní postupy

Celkově lze říci, že rozdílem mezi pojmy být agilní (Be Agile) a používat agilní postupy (Do Agile) je směr, ze kterého se k praktikám a principům přistupuje. Když je tým agilní, tak dochází k využití agilního způsobu myšlení, hodnot a principů, které jsou poté aplikovány pomocí správných praktik. Ty pak mohou být dále upravovány, spojovány či využívány částečně, to záleží jen a pouze na zkušenostech a znalostech týmu. V případě, že tým používá agilní postupy, tak se soustředí hlavně na praktiky a metodiky, které využívá bez znalosti agilního způsobu myšlení. Pro takový tým je pak těžké zjistit kdy, co a jak upravovat v daných metodikách tak, aby tým využíval potenciál agilních metodik na 100 % (Sidky, 2015) (Cardoza, 2015).

Také se od způsobu přístupu k agilním metodikám odvíjí chápání jednotlivých praktik. V případě, že je tým agilní, tak vidí denní standupy jako schůzku, díky které získá nové informace, oproti tomu tým, který „používá agilní postupy“ ho bere jako schůzku, na které se řeší postup vůči plánu sprintu. Dalším příkladem pak může být Scrum Master, který může být vnímán jako člověk, který pomáhá týmu tak, aby uspěl nebo jako projektový manažer, který

hlídá dodržování praktik. Každá praktika může být popsána z obou stran, proto je důležité řádně pochopit dané praktiky a využívat je k tomu, k čemu byly vytvořené a určené (Broza, 2016) (Sidky, 2015).

Na obrázku 3 jsou znázorněny přístupy k agilním metodikám. Kolečka znázorňují jednotlivé praktiky. Trojúhelníky 12 principů z Manifestu agilního vývoje. Čtverce pak čtyři hlavní myšlenky agilního vývoje a zelený mozek je agilní způsob myšlení. Být agilní – agilní přístup jako způsob myšlení a kultura týmu. Používat agilní postupy – agilní přístup jako proces a praktiky (Sidky, 2015).



Obrázek 4 Znázornění rozdílu mezi být agilní a používat agilní postupy, *(Sidky, 2015)*

3 Agilních metodiky a časté chyby při jejich implementaci

V následujícím textu budou stručně stručně představeny některé agilní metodiky a u každé budou uvedeny časté chyby při pokusu o jejich implementaci. Vybranými metodikami jsou Scrum, Extrémní programování a Lean Software Development.

3.1 Scrum

Popis metodiky

Jedná se o framework, který vznikl na počátku devadesátých let a slouží k vývoji komplexního, ale kvalitního softwaru. Umožňuje definovat procesy a techniky pro podporu efektivního vytváření produktu (Schwaber, a další, 2016) (Booz Allen, 2016).

Podstatou metodiky (a základní jednotkou iterace) je tzv. Sprint, což je časový úsek (ne delší než 1 měsíc), který má definovány cíle a jehož výsledkem by měl být funkční přírůstek produktu. Na začátku Scrum team provede sprint planning, jehož výsledkem je výčet úkolů,

které mají být v rámci sprintu splněny. Každý den se provádí 15minutový “standup meeting”, při kterém členové Scrum teamu synchronizují své aktivity a naplánují práci na dalších 24 hodin. Na konci sprintu se provádí sprint review, jež slouží k revizi přírůstku produktu. Před plánováním dalšího sprintu se rovněž doporučuje provést tzv. sprint retrospektivu, během které Scrum team zhodnotí průběh sprintu a případně navrhne, co by se dalo zlepšit nebo dělat jinak. Metodika stojí na třech pilířích (Subramaniam, a další, 2006) (Booz Allen, 2016):

- Transparence – Vývoj musí být viditelný pro všechny účastníky, kteří jsou odpovědní za výsledek. Ti, kdo na projektu pracují, a ti, kdo akceptují výsledky práce, musejí stejným způsobem rozumět používaným termínům. Například musí mít stejnou definici “hotové” práce (tzv. definition of done).
- Inspekce – Uživatelé Scrumu musí často kontrolovat dodržování postupů metodiky a směřování k cíli sprintu, ale tak, aby příliš časté kontroly nebrzdily práci.
- Adaptace – V případě, že se implementace metodiky příliš vychyluje od její definice a ohrožuje kvalitu výsledného produktu, je potřeba vyladit používání metodiky. Za inspekci a adaptaci by měl zodpovídat pověřený pracovník.

Běžné chyby

Uživatelé Scrumu, se velmi často dopouštějí chyb, které obvykle souvisí s přechodem z vodopádového přístupu k vývoji. Mezi tyto chyby patří především (Kingdon, 2016):

- Komplikování implementace Scrumu – Při začátku přechodu na Scrum se týmy snaží používat a přizpůsobit komplikované nástroje. Ty mohou být mocným pomocníkem, jakmile je tým se Scrumem dobře obeznámen a má s ním několikaměsíční praktickou zkušenost, na počátku však naprosto vystačí nástěnka s papírky úkolů nebo tabulka v některém běžném tabulkovém editoru, a manuálně tvořený burndown graf.
- Projekt management – Scrum Master často dělá tu chybu, že se snaží příliš řídit projekt a micromanagovat jednotlivé členy týmu. Scrum Master by však měl být spíše podpůrnou součástí týmu, který by se měl řídit sám. Občas je lepší když se členové týmu poučí z chyb, než aby jim bylo předem řečeno, co přesně a jak, mají dělat.
- Nepřipravený Product Backlog – Vlastník produktu (Product Owner) má zodpovědnost udržovat Product Backlog připravený ještě před začátkem sprintu a obvykle platí, že v průběhu sprintu už by měl být připravený backlog pro další sprint. To je dobrá pojistka proti tomu, že by tým pracoval příliš efektivně a do konce sprintu by neměl co dělat. Připravený backlog pak zajistí, že vývojáři mohou pracovat i na úkolech nad

rámec právě probíhajícího sprintu.

- Špatná komunikace – Členové Scrum týmu se musí naučit komunikovat spolu navzájem, nejlépe osobně. Celý tým velmi zdržuje, pokud jsou členové týmu zvyklí všechno řešit přes Scrum Mastera a mnohdy ještě k tomu elektronicky. Zároveň je potřeba aby vlastník produktu byl vždy k dispozici týmu, připravený zodpovídat dotazy vývojářů během vývoje.
- Špatná denní standup setkání – Zcela běžně se stává, že tato setkání se přestanou zabývat třemi klíčovými otázkami "Co jsem dělal? Co budu dělat dnes? Jaké jsou přede mnou překážky?". Místo toho setkání často sklouzne k plánování práce, rozdělování úkolů a často dokonce k probírání technických detailů řešení konkrétního problému. Indikátorem špatných 15minutových setkání je často to, že netrvají 15 minut, ale třeba 45 minut. Všechny problémy, které nespadají mezi 3 hlavní otázky denního standup setkání by se měly řešit po ukončení tohoto setkání, ne během.
- Neprovádění retrospektivních setkání – tato setkání na konci sprintu jsou důležitá, protože umožňují týmu zlepšit jeho fungování pro další sprint. Velmi často se však stává, že Scrum Master toto setkání "odkládá donekonečna" a nechává jej až na okamžik, kdy bude "více času a méně práce". S tímto přístupem však takový okamžik v agilním vývoji nikdy nenastane.

3.2 Extrémní programování

Popis metodiky

Tato metodika implementuje nejlepší zvyklosti v oblasti programování a dovádí je do extrému. Jedná se především o tyto praktiky (TutorialsPoint, 2015) (Lui, a další, 2006):

- Code review – Kontrola kódu je u extrémního programování prováděna formou párového programování, což je technika, kdy dva vývojáři pracují u jedné stanice s jednou klávesnicí, a to takovým způsobem, že zatímco jeden programuje, druhý ho neustále kontroluje a opravuje.
- Testování – Je zaveden tzv. vývoj řízený testy (Test Driven Development). Při TDD jsou nejprve napsány jednotkové testy a až následně je napsán funkční kód. Testy jsou rovněž pravidelně udržovány a refaktorovány spolu s kódem logiky aplikace.
- Design – Je opuštěna snaha vymyslet perfektní design před začátkem programování, ale místo toho je kód často a pravidelně refaktorován.
- Jednoduchost – Na počátku je zvolen co nejjednodušší design a nejjednodušší kód. V žádném případě není psán kód, který by se mohl později hodit, nebo který bude

potřeba v další iteraci, nýbrž je vždy psáno pouze minimum kódu potřebného pro splnění požadavků aktuální iterace.

- Krátké iterace – Formou plánovací hry (Planning game) společně zákazníci s vývojáři definují sadu požadavků na další iteraci. Je podobná jako sprint planning u Scrumu.

Běžné chyby

- Chyby při párovém programování – Oba vývojáři by se měli pravidelně u klávesnice střídat. Pokud je jeden z nich po několik hodin v roli pozorovatele, může jeho pozornost začít sklouzávat jinam a tato technika je pak zbytečná. Zároveň je potřebná opatrnost, pokud je jeden z obou programátorů značně zkušenější než ten druhý. V takovém případě se může stát, že se ten méně zkušený přesune do pasivní role, kdy pouze poslouchá toho zkušenějšího, a naopak ten zkušenější převezme dominantní postavení a nebude chtít dané řešení se svým partnerem vůbec diskutovat. Párové programování je potřeba dělat správně, jinak hrozí velké plýtvání kapacitou vývojářů.
- Špatná implementace TDD – Kromě nedostatků napsaného kódu testů, je často problém v absenci jejich udržování, při úpravě testovaného kódu, a jejich nedostatečný refaktoring. Dalším běžným problémem je snažit se testy pokrýt příliš mnoho kódu, protože tento přístup může brzdit dodání produktu.

U ostatních principů extrémního programování se také týmy dopouštějí chyb, ty však obvykle plynou jednoduše z absence nebo jen částečného dodržování podstaty daného principu. Například se nesnaží udržet jednoduchost designu (Lui, a další, 2006).

3.3 Lean software development

Popis metodiky

Metodika LSD vznikla převedením Lean metodiky pro výrobu (Lean manufacturing) do prostředí vývoje softwaru. Je založena na 7 principech, které byly více či méně přejaty z verze určené pro řízení výroby (Waters, 2010):

- Eliminace odpadu – V kontextu vývoje softwaru mohou být odpadem například chyby v kódu, nedokončená práce, funkcionalita navíc, byrokracie, nejasné a měnící se zadání.
- Včleňování kvality – Podobně jako Scrum nebo extrémní programování, soustřeďuje se na kontinuální budování kvality technikami jako vývoj řízený testy, párové programování, kontinuální integrace a refaktoring.
- Vytváření znalostí – To obnáší školení mezi kolegy, vytváření znalostní báze a dokumentování různých postupů a kódu.
- Odkládání závazků – Závazná a nezvratná rozhodnutí by měla být odkládána

až na nejzazší možný okamžik. Myšlenka za tímto tvrzením spočívá tom, že by mělo být vyvinuto co nejvíce funkcionalit bez potřeby závazných rozhodnutí, která by mohlo být obtížné později zvrátit. Více rozhodnutí o funkcionalitách by pak mělo být učiněno pospolu.

- Rychlé dodání – Časté a malé přírůstky umožňují získat zpětnou vazbu na produkt od jeho uživatelů co nejdříve a jeho případné úpravy jsou pak nejméně nákladné.
- Respektování lidí – Jejich názory a nápady mohou přispět ke kvalitě produktu, nehledě na jejich roli v teamu. Pravomoc rozhodovat by měla být soustředěna co nejníže v hierarchii a team by se měl podílet na definování požadavků iterací.
- Optimalizace celku – Je potřeba zamezit situacím, které negativně ovlivňují kvalitu kódu – například, když se nové požadavky objeví na poslední chvíli a vývojáři musí pracovat pod silným časovým tlakem (to pak často vede k omezení refaktoringu nebo testování). Činnosti všech zúčastněných stran by měly být sladěny.

Běžné chyby

- Jedním z problémů je, když členové týmu neví o existenci znalostní báze nebo ji nevyužívají, což může být často způsobeno zvolením špatného nástroje (nebo žádného) pro její správu. To pak vede k obtížnému vyhledávání konkrétních informací ve znalostní bázi. Tento problém má za následek, že čas a úsilí, které někteří členové týmu vložili do tvorby znalostní báze, přijde vniveč.

4 Závěr

V této práci byly nejprve představeny hlavní hodnoty a principy agilního vývoje. Na tento základ bylo navázáno podrobným rozborem agilního způsobu myšlení. Nejprve byly popsány způsoby přechodu z tradičních vodopádových metodik k agilním metodikám vývoje. V souvislosti s touto transformací bylo zjištěno, že vývojáři mohou narazit na velké obtíže při snaze přejít na agilní metodiky. Nestačí totiž začít pouze využívat agilní praktiky a řídit se některou z agilních metodik, ale především je potřeba začít agilně přemýšlet a toto přemýšlení dále rozvíjet.

V další části práce byly popsány poznatky psycholožky Carol Dweckové, které rozdělují způsob myšlení na růstové a fixní. Fixní myšlení je více striktní a spíše odpovídá vodopádovému způsobu vývoje, zatímco růstové je blíže nakloněno učení se z chyb a seberozvoji. To je přesně tím, co je potřeba při agilním vývoji. Proto lze růstové myšlení považovat za agilní způsob myšlení. Jak však bylo zjištěno, přechod z jednoho způsobu myšlení na druhý, může být pro členy týmu velmi obtížný.

Práce dále popisuje velmi zajímavou metodu Shu-Ha-Ri vycházející z japonských bojových umění, a její adaptaci na agilní myšlení.

V závěrečné kapitole práce byly stručně popsány metodiky agilního vývoje Scrum, extrémní programování a Lean Software Development a některé z častých chyb, kterých se týmy dopouštějí při snaze tyto metodiky implementovat.

Poznatkem práce je především to, že je rozdíl mezi pojmy „používat agilní postupy“ a „být agilní“. Ani nejpřesnější dodržování pravidel metodiky totiž nestačí k tomu, aby z ní plynula přidaná hodnota, pokud se lidé nenaučí agilně myslet. Proto není až natolik důležité umět přesně implementovat nějakou metodiku, ale spíše pochopit principy, na kterých jsou různé metodiky postaveny. Týmy si pak budou schopny použité metodiky přizpůsobit nebo je dokonce vhodným způsobem zkombinovatm, a tím ještě více zvýšit svoji efektivitu.

5 Reference

Agile-Scrum. 2016. What is the Agile Mindset? *Agile-Scrum.* [Online] 2016. [Citace: 16.. 4. 2017.] http://www.agile-scrum.be/agile-software-development/agile-mindset/.

Beck, Kent, a další. 2001. Manifest Agilního vývoje software. *agile manifesto.* [Online] 2001. [Citace: 15. 4. 2017.] http://agilemanifesto.org/iso/cs/manifesto.html.

Booz Allen. 2016. Agile playbook. *Boozallen.* [Online] 9. 6. 2016. [Citace: 1. 3. 2017.] https://github.com/booz-allen-hamilton/agile-playbook.

Broza, Gil. 2016. The mindset of the agile developer. *Better Software.* 2016, Sv. FALL, str. 20-23.

Cardoza, Christina. 2015. Don't do agile, be agile. *SDTimes.* [Online] 1. 9. 2015. [Citace: 16. 4. 2017.] http://sdtimes.com/dont-do-agile-be-agile/.

Dak, Ruma. 2013. Shu-Ha-Ri and AGILE. *Ruma Dak's Blog.* [Online] 21. 9. 2013. [Citace: 16. 4. 2016.] https://rumadak.wordpress.com/2013/09/21/shu-ha-ri-and-agile/.

Denning, Steve. 2016. What's Missing In The Agile Manifesto: Mindset. *Forbes.* [Online] 7. 6. 2016. [Citace: 15. 4. 2017.] https://www.forbes.com/sites/stevedenning/2016/06/07/the-key-missing-ingredient-in-the-agile-manifesto-mindset/.

Easuwaran, Sathish. 2017. Developing an Agile Mindset. *Saksoft.* [Online] 27. 2. 2017. [Citace: 16. 4. 2017.] https://www.saksoft.com/developing-an-agile-mindset/.

Fowler, Martin. 2014. ShuHaRi. *Martin Fowler.* [Online] 22. 8. 2014. [Citace: 16. 4. 2017.] https://martinfowler.com/bliki/ShuHaRi.html.

Howard, Leanne. 2015. What Does It Mean to Have an Agile Mindset? [Online] 1. 4. 2015. [Citace: 15. 4. 2017.]

Jeremiah, John. 2016. Survey: Is agile the new norm? *TechBeacon.* [Online] 2016. [Citace: 15. 4. 2017.] https://techbeacon.com/survey-agile-new-norm.

JustSolve. 2016. The Agile Mindset. *JustSolve.* [Online] 11. 9. 2016. [Citace: 15. 4. 2017.] http://www.justsolve.co.za/agile-mindset/.

Kingdon, Dwight. 2016. 10 Common Scrum Mistakes and How to Avoid Them. *DZone.* [Online] 8. 8. 2016. [Citace: 16. 4. 2017.] https://dzone.com/articles/10-common-scrum-mistakes-and-how-to-avoid-them.

Kolektiv autorů. 2016. In a nutshell, why do a lot of developers dislike Agile? *Quora.* [Online] 2016. [Citace: 1. 3. 2017.] https://www.quora.com/In-a-nutshell-why-do-a-lot-of-developers-dislike-Agile.

Lui, Kim Man a Chan, Keith C.C. 2006. Pair programming productivity: Novice–novice vs. expert–expert. *UTexas.* [Online] 21. 4. 2006. [Citace: 16. 4. 2017.] http://www.cs.utexas.edu/users/mckinley/305j/pair-hcs-2006.pdf.

Mysliveček, Ondřej. 2011. Agilní vývoj v praxi. *SystemOnLine.* [Online] 24. 9. 2011. [Citace: 15. 4. 2017.] https://www.systemonline.cz/sprava-it/agilni-vyvoj-v-praxi.htm.

Novack, Jason. 2016. Shu Ha Ri: An Agile Adoption Pattern. *SolutionsIQ.* [Online] 26. 5. 2016. [Citace: 16. 4. 2017.] http://www.solutionsiq.com/shuhari-agile-adoption-pattern/.

Popova, Maria. 2015. Fixed vs. Growth: The Two Basic Mindsets That Shape Our Lives. *brainpickings.* [Online] 18. 9. 2015. [Citace: 16. 4. 2017.] https://www.brainpickings.org/2014/01/29/carol-dweck-mindset/.

Schwaber, Ken a Sutherland, Jeff. 2016. The Scrum Guide. *scrumguides.* [Online] 1. 7. 2016. [Citace: 15. 4. 2017.] http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100.

Sidky, Ahmed. 2015. The agile mindset. *Softed.* [Online] 14. 9. 2015. [Citace: 15. 4. 2017.] https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf.

Subramaniam, Venkat a Hunt, Andy. 2006. *Practices of an Agile Developer.* Raleigh : Pragmatic Bookshelf, 2006. ISBN 0-9745140-8-x.

Šochová, Zuzana. 2013. Shu-Ha-Ri aneb návrat k základům. *Soch.* [Online] 2. 11. 2013. [Citace: 16. 4. 2016.] http://soch.cz/blog/management/agile/shu-ha-ri-aneb-navrat-k-zakladum/.

TutorialsPoint. 2015. Extreme Programming - Introduction. *tutorialspoint.* [Online] 2015. [Citace: 15. 4. 2017.] https://www.tutorialspoint.com/extreme_programming/extreme_programming_introduction.htm.

Waters, Kelly. 2010. 7 Key Principles of Lean Software Development. *All About Agile.* [Online] 16. 8. 2010. [Citace: 15. 4. 2017.] http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/.

6 Seznam obrázků