Semestrální práce ke kurzu 4IT421 – Zlepšování procesů budování IS	
Semestr	LS 2016/2017
Autoři	Marcel Danilov – xdanm33 Anton Mashera – xmasa00
Téma	Actionable Agile Tools
Datum odevzdání	14. 5. 2017

Abstrakt

Téma této práce jsou tzv. *actionable* agilní techniky. Pod pojmem *actionable* si lze představit techniky, které jsou praktické a osvojitelné v relativně krátkém čase. Laicky řečeno jsou to techniky, nad kterými lze okamžitě jednat (anglicky *take action*), a proto se označují jako *actionable*.

Práce je rozdělena do několika kapitol. První kapitola slouží jako úvodní představení problematiky, je zde obecně představen pojem agilní vývoj, metodiky, techniky atd. Dále je zde uveden autor zdrojového článku Jeff Campbell, z jehož hlavy celá idea *actionable* agilní technik pochází. Druhá kapitola pojednává o problematice implementace agilních metodik. Je zde uvedeno několik problémů, na které mohou uživatelé narazit. Třetí kapitola se týká samotných *actionable* agilní technik. Několik jich je zde představeno.

Cílem práce je poskytnout přehledný soupis několika technik, které by měly uživatelům pomoci ve fázi implementace agilního vývoje. Zmíněné techniky by měly poskytnout určitý startovní bod, od kterého se lze v úvodních fázích implementace odrazit a vytvořit tak pevný základ jak pro proces implementace, tak užívání agilního vývoje v budoucnu. Dílčím cílem je současně i upozornění na některé problémy, na které mohou týmy při implementaci agilních metodik narazit.

Klíčová slova

agilní vývoj, agilní techniky, agilní metodiky, implementace

Obsah

1 Úvod

1.1 Agilní vývoj a metodiky

Ústředním tématem této práce je způsob vyvíjení softwaru a metodiky při tom užívané, které se označují jako agilní nebo také lehký. Proti agilnímu vývoji existuje rovněž přístup protichůdný, který se označuje jako tradiční (rigorózní nebo těžké). Pro lepší náhled do problematiky agilního přístupu je nutné definovat nejen samotný agilní, ale i již zmíněný opačný tradiční přístup.

Agilní vývoj je nejlépe definovatelný podle principů, které ctí. Principy agilní vývoje jsou definovány v takzvaném „*Agilním Manifestu*“ a jsou následující (Agilní Manifest, 2001):

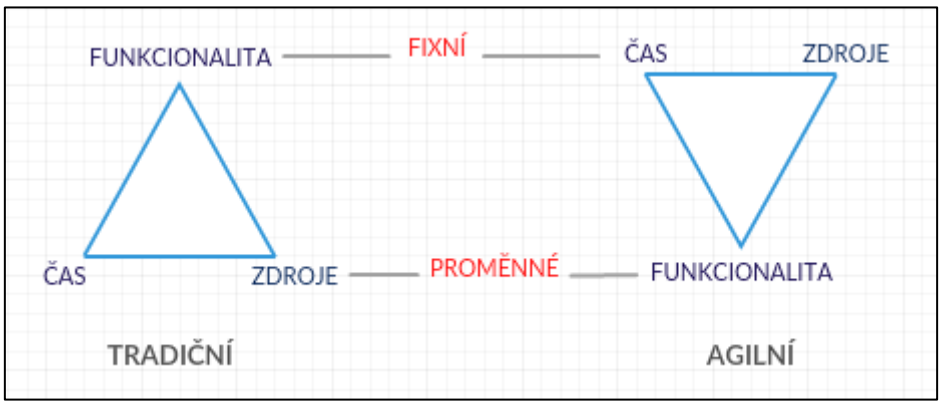
- *JEDNOTLIVCI A INTERAKCE* před procesy a nástroji
- *FUNGUJÍCÍ SOFTWARE* před vyčerpávající dokumentací
- *SPOLUPRÁCE SE ZÁKAZNÍKEM* před vyjednáváním o smlouvě
- *REAGOVÁNÍ NA ZMĚNY* před dodržováním plánu

Z toho vyplývá, že primární zaměření agilních metodik je na přizpůsobení se zákazníkovi, jeho zatažení do procesu vývoje, neustále upřesňování požadavků a následná flexibilní reakce. Proces vývoje je realizován přírůstkově (inkrementálně) a v cyklech (v iteracích). Zákazník mnohdy v začátcích nemá jasnou představu o finálním produktu, spíše jen hrubý odhad a jeho finální podoba tak dostává konkrétní obrysy až v samotném procesu vývoje.

Tradiční přístup je nejlépe definován tzv. Vodopádovým modelem. Tento model je sekvenční (neiterativní) proces vývoje rozdělený do několika fází jako specifikace požadavků, analýzy, návrhu, implementace, testování a zavedení. V první fázi lze nalézt velký rozdíl oproti agilnímu přístupu. Specifikace požadavků probíhá na začátku celého procesu. Idea je, že nelze tvořit, vyvíjet nebo budovat, dokud nejsou formulovány všechny požadavky v kompletní podobě a nejsou náležitě zdokumentovány. Dalším znatelným rozdíl je samotné zapojení zákazníka, neboť ten se do vývoje zapojuje pouze na začátku a konci celého procesu a v průběhu tak nemá žádnou kontrolu nad projektem.

Rozdíl mezi těmito přístupy lze snadno ukázat i na trojúhelníku s vrcholy Zdroje, Čas a Funkcionalita. Agilní přístup určuje v průběhu projektu veličiny Zdroje a Čas jako fixní čili neměnné a veličiny Funkcionalita jako proměnnou. Zjednodušeně to znamená, že na začátku projektu se pevně naplánují čas a zdroje a samotná funkcionalita je volně upravována až v průběhu projektu.

Tradiční přístupy na druhou stranu označují veličinu Funkcionalita jako fixní a Čas a Zdroje jako proměnné. To znamená, že na začátku projektu se na pevno určí výsledná funkcionalita a se zdroji a časem se poté hýbe dle potřeby, tzn. projekt může ve výsledku stát více a trvat mnohem déle, než bylo naplánováno, nicméně se dosáhne požadované funkcionality. Trojúhelníku obou přístupů jsou znázorněné na obrázku níže.



Obrázek 1 - srovnání přístupů podle trojúhelníků, zdroj: Autor

1.2 Jeff Campbell, autor Actionable Agile Tools

Jak již je zmíněno výše, tématem této práce jsou tzv. *actionable* agilní techniky. S ideou těchto technik přišel Jeff Campbell, Kanaďan, v současné době žijící ve Švédsku. Campbell se živí jako tzv. agilní kouč. Pod touto profesí si lze představit člověka, který společnostem jak malým, tak velkým, pomáhá s přechodem na agilní vývoj, funguje zde jako mentor, či konzultant.

V současné době Jeff Campbell pracuje ve společnosti *Meltwater*, kde jeho hlavní zodpovědnost leží na pozici *Scrum Mastera*. V minulosti Campbell vypomáhal i společnostem jako *HP, Volvo či Lavasoft*. Campbell je v komunitě agilního vývoje výrazně angažován i mimo své pracovní prostředí. Je jedním ze zakladatelů agilní komunity ve švédském Göteborgu s názvem *Scrum Beers*. Současně pořádá i konference s tématem agilního vývoje s názvem *Brewing Agile*.

2 Problémy při implementaci agilních metodik

Ve zdrojovém článku s názvem Actionable Agile Tools (Campbell, 2016) zmiňuje autor Jeff Campbell jeden z problémů, na který za svoji kariéru agilního kouče narazil. Tímto problémem je, že při přechodu na agilní metodiky se často naráží na problém, že tyto metodiky nejsou popsané s dostatečnou hloubkou či vážností a lidé tak nevědí, kde při přechodu na agilní metodiky prakticky začít, od čeho se v počátcích odrazit, jednoduše řečeno jim chybí určitý „startovní bod“. Tento problém Campbell v článku řeší popisem praktických (*actionable*) agilních technik. Některé z nich jsou popisovány v kapitole 3. Zmíněný problém však určitě není jediný problém, na který lze při přechodu na agilní metodiky narazit. Následující stránky jsou proto věnovány několika dalším problémům.

Popisované problémy jsou rozděleny do několika skupin podle toho, čeho se týkají. První tři problémy se týkají „lidského faktoru“ poněvadž lidé a lidská interakce tvoří jádro agilních metodik (dokumentováno např. Agilním Manifestem, kde jeden z principů je JEDNOTLIVCI A INTERAKCE před procesy a nástroji). Dalších několik problémů je spíše technického zaměření.

2.1 Strach vývojářů způsobený transparentností práce

(Olesiejuk, 2011) říká, že progres práce každého člena týmu je většinou dokumentován či reportován na denní bázi, na různých poradách, stand-up mítincích atd. Díky tomu má každý člen týmu přehled o tom, jakou práci vykonávají ostatní členové a jak dlouho jim daná práce trvá. Právě délka práce je zde důležitá, neboť pokud určité osobě trvá daný úkol déle, než by měl, tak daná osoba může mít, že ji ostatní za zády soudí, mluví o tom, proč daný úkol trvá tak dlouho atd. Transparentnost práce může dále „odhalit“ určité nedostatky v technických znalostech či schopnosti komunikovat. Obecně poté platí, že díky těmto problémům způsobeným transparentností, mohou mít někteří lidé problémy se sebevědomím.

Prevence proti těmto problémům by mohla být následující. Členové týmu by se měli cítit bezpečně, co se týče odhalování svým slabých stránek, problémů při práci atd. Dobrým nápadem může být např. pořádání menší porady po hlavní poradě, kde bude přítomna menší skupina lidí. Někteří se v této menší skupině mohou cítit otevřenější. Dalším řešením může být např. konstantní poskytování příležitostí, kde jednotliví členové týmu mohou své znalosti či schopnosti vylepšovat. Členové týmu na pozici na „junior“ by měli mít svého mentora, kterému se mohou se svými problémy svěřovat a který jim bude na denní bázi pomáhat.

Excelentní technikou je rovněž párování. Zkušenější a služebně starší tak mohou své poznatky, zkušenosti, znalosti sdílet s těmi méně zkušenými.

2.2 Potřeba aby vývojář uměl vše

(Olesiejuk, 2011) rovněž zmiňuje další problém, kterým je potřeba, aby vývojář byl znalý v široké oblasti problematik. Problém je uvozen citátem neznámého manažera: „Pro to, abyste byl úspěšným agilním vývojářem musíte být najednou programátor, tester, architekt, zákazník, kontrolor kvality a mnoho dalších věci spojených s vývojem SW“. Zmíněný citát je pravdivý, nicméně problém nastává v komplexních projektech. Jak jeden člověk může současně zvládnout pozici testera a např. architekta nebo databázového experta? Odpověď je jednoduchá. Nemůže. Platí zde staré pořekadlo „Devatero řemesel, desátá bída.“

Mnoho společností se pokouší své zaměstnance posílat na velké množství tréninků, seminářů atd. Nicméně tyto pokusy o školení zaměstnanců v široké oblasti problematiky se ukázaly jako drahé a neefektivní. Řešení tohoto problémů není evidentní, nicméně existuje. Je potřeba najít rovnováhu mezi obecnými znalostmi a jednotlivými specializacemi. Vedoucí týmu by měl své členy vybírat pečlivě. Vývojáři by měli obecné znalosti v oblasti SW vývoje a současně i business stránky, nicméně by stále měli zůstávat odborníky v určité oblasti (výše zmíněný programátor, tester atd.). I přesto však zde zůstává problém s velikostí týmu. V malých týmech je toto řešení mnohem lépe realizovatelné než ve velkých, komplexních týmech.

2.3 Zvýšená závislost na sociálních dovednostech

Dalším problémem zmíněným v (Olesiejuk, 2011) je zvýšená závislost na sociálních dovednostech. Díky konstantní komunikaci v agilních metodikách by členové týmu měli mít dobré sociální či komunikační dovednosti. Celkem často nastává problém, že v týmu existuje kvalitní vývojář, který však pokulhává v oblasti komunikace. To představuje velký problém, neboť pro dané členy je obtížné vyjádřit zbytku týmu své myšlenkové pochody, nápady atd. Dalším problém může nastat, pokud se musí komunikovat se zákazníkem. Členové týmu jako vývojáři často nemusí mít nutně problémy v komunikaci mezi sebou, neboť se znají a delší dobu spolu již pracují, nicméně problémy mohou nastat právě v komunikaci s klientem. Podstatným elementem agilních metodik totiž je, že např. vývojáři musí o požadavcích na určité funkcionality právě hovořit přímo s klienty.

Nejvíce intuitivní řešení zde je zaslání členů týmu na nějaký druh semináře, kde budou mít možnost své sociální dovednosti zlepšovat. Toto řešení je dle některých průzkumů nejvíce efektivní. Dalším dobrou praktikou může být např. nahrávání mítinků a jejich následná analýza. Dále kontakt s klientem by nutně nemusel být realizován individuálně, nýbrž ve skupině, kde se jednotlivé nedostatky v komunikaci mohou ztratit. Poslední rada je spíše preventivní než reaktivní – při sestavování agilního týmu najímat pouze lidi, kteří už určitou úroveň komunikačních dovedností mají.

Jak je již zmíněno v úvodu této kapitoly, další problémy, které zde budou popisovány se již netýkají „lidského faktoru", nýbrž se více zaměří na technickou a organizační stránku věci. Zmíněné problémy jsou popisovány v (Agile Implementation Problems, 2016).

2.4 Špatné user stories

Prvně je třeba definovat co vůbec „user story" (uživatelský scénář) znamená. (Agile v kostce, 2016) user story definuje jako „definici požadavků s akceptačními kritérii od dané osoby". Problém s těmito uživatelský scénáři je podle (Agile Implementation Problems, 2016), že tyto scénáře jsou špatně napsané – nedávají smysl nebo jsou špatně formulované. Řešením je vzájemná spolupráce s týmem, který na scénáři bude pracovat, už v době psaní takového scénáře.

2.5 Nesprávné prioritizování

Správné určení priorit se může zdát jako poměrně jednoduchý úkon, nicméně překvapivě velké množství týmů s tím má problémy a nezvládá dobře prioritizovat. Řešení může být delegování rozhodovacích pravomocí členům týmu, kteří na projektu opravdu pracují (oproti možnosti kdy „decision making" – dělání rozhodnutí je výhradně na manažerovi). Tito členové si sami dokáží určit položky s vysokou prioritou, na kterých se bude pracovat a současně určit položky s nižší prioritou, které mohou být dočasně pozastaveny.

3 Actionable Agile Tools

Tématem této kapitoly jsou již jednotlivé „actionable“ agilní techniky. Je jich zde popsáno hned několik, konkrétně 4. Výklad je doplněn obrázky. Zdrojem je (Campbell, 2015).

3.1 Door Calendar

Při vývoje softwaru se zpravidla stává, že všechny myšlenky jsou soustředěny na dodání finálního produktu a lze tak snadno ztratit ze zřetele důležité věci a události, které se blíží. Existují také události, které se nastávají v nepravidelných intervalech jako svátky či firemní akce, s nimiž je potřeba také počítat a pamatovat si je. K tomu abyste měli přehled o všem, co Vás čeká v nejbližší době lze použit nástroj *Door calendar*.

Door calendar čili dveřní kalendář, byl tak pojmenován, protože dveře jsou zpravidla nejlepším místem, kam lze takový kalendář umístit, jelikož jsou často používány všemi. Jedná se o jednoduchý informační nástroj, který umožňuje mít přehled o tom, co se v nejbližším období plánuje udělat nebo na kterou událost je nutno se připravit. Na obrázku 2 je znázorněn příklad, jak lze takový kalendář sestavit.

Obrázek 2 – Door Calendar, zdroj: (Campbell, 2015)

Další výhodou tohoto nástroje je, že jeho konstrukce velice jednoduchá. Stačí Vám lepící páska, marker a poznámkové lístky a samozřejmě dveře, na které tento kalendář umístíte. Důležité je, aby velikost buněk byla dostatečná na to, aby tam pohodlně šlo umístit poznámkové lístky. Na samotném papíru pak stačí krátce popsat událost, která se blíží. Jako indikátor dnešního dne lze například použit další lístek s nakreslenou šipkou, jak je to zobrazeno na obrázku 2. Nahoře nad poznámkovými lístky se píšou dny v týdnu a vlevo před začátkem týdne je umístěn další lístek na kterém je napsáno číslo týdne nebo datum (např. 03.04 – 07.04).

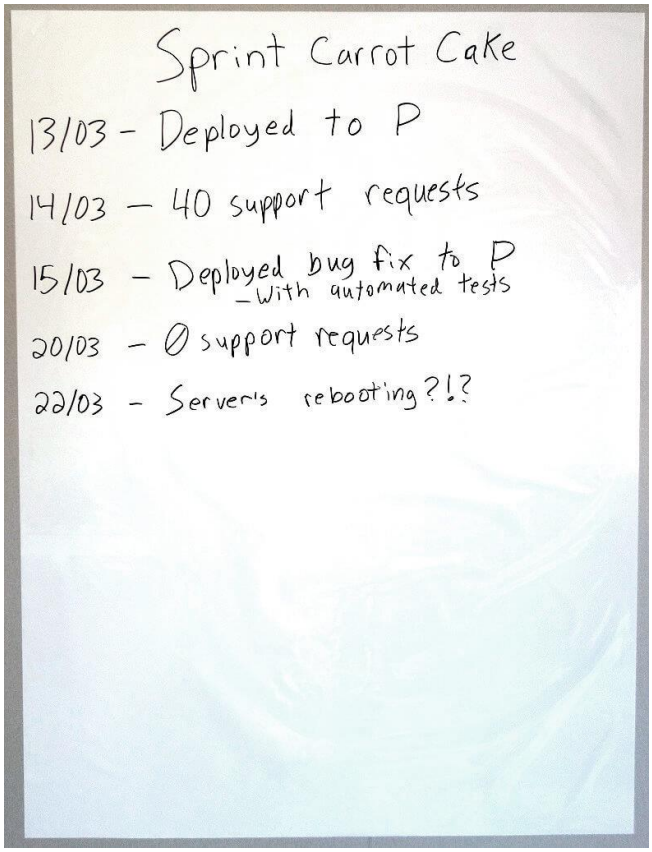
Během každodenních meetingů se odstraňuje dnešní poznámka a do kalendáře jsou přidávány nové události, které byly domluveny na schůzce. Na začátku každého týdne celý kalendář se posune o jeden řádek nahoru a dolu se nalepí další týden. Upravovat a doplňovat dveřní kalendář je dobré spolu se svými kolegy z týmu, aby všichni věděli o případných změnách a aby se nikomu nic neuniklo.

Aby *Door calendar* přinášel maximální užitek je potřeba dodržovat následující pravidla:

- Když odstraňujete starou poznámku, případně přidáváte novou, snažte se, aby to viděli všichni členové týmu.
- Je dobré používat různobarevné lístečky pro různé typy událostí a legendu, ve které bude uvedeno, co která barva znamená (např. růžová znamená volný den atd.).
- Nestydět se dávat do kalendáře zábavné věci, jako například výlety a týmové akce, jak je uvedeno na obrázku 2.
- Psát datum do lístků s čísly týdnů.
- Nedělat kalendář zbytečně moc komplikovaným.
- Dávat do kalendáře také víkendy, sice s velkou pravděpodobností nebudou použity pro poznámky, ale lépe se na to dívá.

3.2 Event Log

Dalším velkým problémem při agilním vývoji je to, že se velice často stává, že si lidé těžko vzpomínají, co dělali posledních několik dnů nebo týdnů. Například mohou zapomenout na důležité věci, které se probírali na nějaké předchozí denní poradě. Řešením tohoto problému může být nástroj *Event log*. Jeho konstrukce je ještě jednodušší než u *Door calendar* – všechno co potřebujete je velký kus papíru, nejlépe velikosti A1 a fixy. Na každodenních schůzkách je pak třeba najít si čas napsat na tento papír všechny věci z předchozí porady, které všichni považují za důležité, aby se pamatovaly. Na obrázku 3 je znázorněno, jak by mohl vypadat *Event log*.



Obrázek 3 - Event Log, zdroj (Campbell, 2015)

Letmý pohled na *Event Log* stačí na to, aby si lidé připomněli události, které by jinak byly zapomenuty a nikdy se o nich nediskutovalo. Výhodou je, že *Event log* nám umožňuje se soustředit na předchozí den před tím, než se pustíme do dne následujícího.

Tipy pro sestavení *Event logu*:

- Aktualizovat *Event Log* každý den. Pokud jej budete aktualizovat kdykoli se Vám bude chtít, pak určitě přijdete o důležité věci.
- Do *Event logu* lze také zapsat i úspěšně vyřešené problémy ve formě seznamu, je příjemně vidět i úspěchy, a nejen nevyřešené úlohy.
- Lze na něm shromažďovat i další data, jako jsou například počet nevyřešených problémů, spokojenost týmu (např. ve formě smajlíků), důvěra v současný plán atd. Nicméně se doporučuje moc nepřehánět, protože poté se *Event Log* může stát moc nepřehledným.

3.3 Resilience Map

Při agilním vývoji se lze setkat s cross-funkčními týmy a T-shaped profesionály. T-shape profesionál je člověk, jehož znalostní profil má tvar T. Do hloubky směřují jeho znalosti IT, do šířky rozhled v různých oblastech, které zahrnují marketing, management, ekonomiku, soft skills a další. Díky tomu pak může analyzovat problém a navrhnout vhodné IT řešení. Dalším charakteristickým rysem agilního vývoje jsou nejednoznačně stanovené kompetence, lidé mohou pracovat zároveň na několika úkolech a vykonávat různé role. Existuje snaha nenechávat kompetenci na jedinci, jelikož lidé odcházejí a přicházejí. Kvůli tomu pak může vzniknout problém s nedostatečnou kapacitou – člověk, který je schopen daný úkol vykonat pracuje na něčem jiném a z toho důvodu jsme vynucení začít pracovat na jiném úkolu, který má nižší prioritu. Dalším problémem mohou být obavy z odchodu kvalifikovaného personálu. Je obtížné mít povědomí o úrovni odborných znalostí celého personálu a kvůli tomu pak může vzniknout strach, že pokud odejde nějaký zkušený člen týmu, tým nebude schopen spravovat nějakou část systému, protože neví, kdo by mohl odchozího člena nahradit. Pro lepší přehled o odborných znalostech a schopnostech personálu je dobrá *Resilience Map*, jenž zobrazena na obrázku 4.

Obrázek 4 – Resilience Map, zdroj (Campbell, 2015)

Jedná se o informační nástroj, který vytváří tým sám. *Resilience Map* lze vytvořit na jedné z tabulí v kanceláři. Budete k tomu potřebovat poznámkové lístky, fixy a obrázky panáčků se jmény členů týmu. Na ose X se uvádějí technické dovednosti nebo oblasti vytvářeného produktu např. UI, DB atd. Na ose Y se uvádí tzv. úroveň komfortu, tj. úroveň porozumění problému (None, Basic, Medium, High). Panáčci v buňkách matice reprezentují jednotlivé členy týmu. Při pohledu na *Resilience Map* je tak hned vidět v jaké oblasti existují problémy a v jaké naopak máme dostatečnou rezervu. Tím lze jednodušeji stanovit priority. Například z obrázku 4 lze posoudit, že tým má problém v oblasti plateb, a naopak všichni dobře rozumí grafickému rozhraní.

Tipy pro práci s *Resilience Map*:

- Je užitečné brát si ji s sebou na plánovací schůzky, protože to pomůže zhodnotit nedostatky v jednotlivých oblastech a jejich možné účinky.
- Používat ji jako vysvětlení pro klienta a stakeholdery proč zrovna tento úkol má vyšší prioritu.
- Nedávat do mapy úplně všechny oblasti. Nebude moc čitelná a bude těžká na udržování. Je lepší se soustředit na věci s vysokou prioritou.
- Vybrat si pravidelný interval, ve kterém bude mapa aktualizována, přičemž je dobré aktualizovat nejen panáčky, ale také pole podél osy Y.
- Zvážit, zda organizace je připravena na tuto úroveň otevřenosti. Pokud je vidět, že někdo v týmu je z toho trošku nervózní, pak je lepší si to nechat jako soukromý nástroj.

3.4 Tasty Timeboxes

Během vývoje softwaru s použitím agilních metodik, často existuje potřeba se odkázat na poslední sprint nebo iteraci. Například při diskuzi o tom, co bylo vydáno nebo pokud je potřeba připomenout týmu nějakou specifickou situaci z minulosti, z předchozího sprintu atd. Pro tyto účely existuje více možností: očíslovávat sprinty, pojmenovávat podle cílů sprintu nebo lze vybrat nějaký motiv (např. nějaký chemický prvek) a sprint pojmenovat podle něj.

Mnohem zábavnějším řešením jsou *Tasty Timeboxes*. Funguje to obdobně jako při pojmenovávaní sprintu, u prvního sprintu název začíná písmenem A a dále pak podle abecedy. V tomto případě ale samotný název znamená nějaké jídlo nebo pochoutku, kterou Váš tým má rád. Mohou to být například pečené dobroty jako: ***A**pple pie, **B**lueberry pie, **C**heesecake atd.* Na *Sprint Review*, na konci každého sprintu je poté možné danou pochoutku donést. Kromě důležité diskuze během *Sprint Review* se tak členové týmu spolu dobře najedí. A nemusí to být

pouze pekařské výrobky, lze použít i něco jiného, např. ovoce, zmrzlinu, bonbony atd. Hlavně, aby to všichni měli rádi. Členové týmu tak mají možnost nejen probrat, jak kvalitní a funkční software vyvinuli, ale dát si spolu něco k jídlu, co bude chutnat všem. Takový přístup podporuje neformální komunikaci a je mnohem zábavnější než obyčejné pojmenovávání sprintů.

Existuje několik doporučení, jak správně tento nástroj používat v praxi:

- Je potřeba vybrat něco, co všichni z týmu mají rádi.
- Pokud dojde ke hlasování o tom, co by to mělo být, je potřeba si vybrat jednoduchý systém hlasování.
- Po nějaké době může být problém sehnat dobrovolníka, který by například upekl nějaký koláč. Někteří lidé budou mít pocit, že jsou to vždy oni, kdo něco dělá, zatímco ostatní nedělají nic. Jednodušší variantou v tomto případě bude jenom si něco koupit v obchodě.

4 Závěr

Cílem této práce s názvem Actionable Agile Tools bylo poskytnout čtenáři přehledný soupis několika technik, které mají za úkol uživateli zjednodušit proces fáze implementace agilních metodik, poskytnout uživateli určitý startovní bod. V této práci byly popsány celkem 4 takové techniky (kapitola 3), a to Door Calendar, Event Log, Resilience Map a Tasty Timeboxes. Cíl práce tedy lze označit jako splněný, nicméně zmíněné 4 techniky představují pouze dílčí část práce Jeff Campbella, autora zmíněných technik. Další techniky jsou popsány v článku (Campbell, 2016) a knize (Campbell, 2015). Pro hlubší studování této problematiky je nutné, aby si čtenář nastudoval i zmíněné zdrojové prameny, a ne pouze tuto práci. Dílčím cíle této práce bylo současně i upozornění na některé problémy, na které lze narazit při implementaci agilních metodik. I tento cíl je považován za splněný. V kapitole 2 bylo popsáno 5 zmíněných problémů.

Použité zdroje

Agile Manifesto [online]. 2001 [cit. 2017-03-29]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

CAMPBELL, Jeff. Actionable Agile Tools. *InfoQ* [online]. 2016, , 1 [cit. 2017-05-13]. Dostupné z: https://www.infoq.com/articles/actionable-agile-tools

CAMPBELL, Jeff. *Actionable Agile Tools* [online]. 2015 [cit. 2017-05-14]. Dostupné z: https://leanpub.com/actionableagiletools

OLESIEJUK, Pawel. Key Challenges in Agile Implementations. *Goyello* [online]. 2011, , 1 [cit. 2017-05-13]. Dostupné z: https://blog.goyello.com/2011/11/28/key-challanges-in-agile-implementations/

Agile Implementation Problems. *Kanban Tool* [online]. 2016 [cit. 2017-05-14]. Dostupné z: http://kanbantool.com/blog/agile-implementation-problems

Agile v kostce. *Agile Ostrava* [online]. 2016 [cit. 2017-05-14]. Dostupné z: http://agileostrava.cz/cs_CZ/agile-v-kostce/