Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Diplomová práce

Gamifikace jako technika k zefektivnění Scrumu ve vybrané společnosti

Bc. Miloslav Haba

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Miloslav Haba

Projektové řízení

Název práce

Gamifikace jako technika k zefektivnění Scrumu ve vybrané společnosti

Název anglicky

Gamification as a technique for streamlining Scrum in selected company

---

Cíle práce

Účelem práce je navrhnuti gamifikačních prvků ke zdokonalení aplikace agilní metodiky Scrum u konkrétní firmy. Cíl práce se skládá z následujících bodů.

1. Prostudování historie agilních metodik.
2. Porovnání agilního přístupu s tradičním pojetim projektového řízení.
3. Vymezení jednotlivých agilních metodik, především metodiky Scrum.
4. Definování gamifikace a rozbor praktického užiti gamifikačních prvků.
5. Zpracování návrhu implementace gamifikace na vybraném a popsaném projektu ve zvolené společnosti.
6. Doporučení dalšího postupu/výzkumu.

Metodika

Práce čerpá z teoretických základů agilních metodik, konkrétně pak Scrumu. K tomu definuje pojem gamifikace a jmenuje příklady jejího uplatnění. V obou případech vychází ze studia odborné literatury.

Na základě polostandardizovaného rozhovoru se Scrum masterem přináší ukázku Scrumu v praxi. Zjištěné skutečnosti porovnává s teoretickou bází. Poté uvádí možnosti zavedení gamifikačních prvků ke zlepšení využiti této metodiky v dané společnosti.

Kvantitativní složka práce je zastoupena příslušnými grafy a tabulkami.

Doporučený rozsah práce

60-80 stránek

Klíčová slova

Projektové řízení, Agilní metodiky, Scrum, gamifikační prvky

Doporučené zdroje informací

BUCHALCEVOVÁ, A. *Metodiky vývoje a údržby informačních systémů : kategorizace, agilní metodiky, vzory pro návrh metodiky.* Praha: Grada, 2005. ISBN 80-247-1075-7.

LACKO, B. – DOLEŽAL, J. – MÁCHAL, P.– SPOLEČNOST PRO PROJEKTOVÉ ŘÍZENÍ. *Projektový management podle IPMA.* Praha: Grada, 2012. ISBN 978-80-247-4275-5.

SCHWABER, K. – BEEDLE, M. *Agile Sofware Development with Scrum.* Upper Saddle River: Prentice Hall, 2002. ISBN 0-13-067634-9.

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Petra Pavlíčková, Ph.D.

Garantující pracoviště

Katedra systémového inženýrství

Elektronicky schváleno dne 22. 11. 2018

doc. Ing. Tomáš Šubrt, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 29. 11. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 29. 03. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Gamifikace jako technika k zefektivnění Scrumu ve vybrané společnosti" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 29. 3. 2019 _____

Poděkování

Rád bych touto cestou poděkoval paní Ing. Petře Pavlíčkové, Ph.D., za ukázkové vedení mé diplomové práce. Dále bych rád zmínil také ty, kteří stojí za každých okolností při mém boku – jejich podpora je mi velkou motivací.

Gamifikace jako technika k zefektivnění Scrumu ve vybrané společnosti

Abstrakt

Účelem práce je navrhnutí gamifikačních prvků ke zdokonalení aplikace agilní metodiky Scrum u konkrétní firmy, přičemž se tento cíl práce skládá ze šesti samostatných bodů. Práce se zabývá jak historickým kontextem agilních metodik, tak podrobným srovnáním modelu Waterfall s paradigmatem Agilu. Následně čerpá z teoretických základů agilních metodik, zejména pak Scrumu. Důraz je v daném ohledu kladen na studium odborné literatury. Praktická část směřuje k vlastní formulaci gamifikace a rozboru tematicky blízkých odborných studií. Poté prezentuje zhodnocení polostandardizovaného rozhovoru se Scrum masterem, na jehož základě je navržen k implementaci příhodný gamifikační koncept.

Klíčová slova: Projektové řízení, Agilní metodiky, Scrum, gamifikační prvky, rámec Scrumu, Scrum Poker, historie Agilních metodik, Waterfall, Agile manifesto, Scrum master

Gamification as a technique for streamlining Scrum in selected company

Abstract

The purpose of thesis is to design gamifying elements in order to improve the application of Scrum agile methodology for selected company. The main goal is divided into six separated parts. First part is about historical context of agile methodologies, whilst second includes comparison of the Waterfall model with paradigm of Agile. There are also shown theoretical basics of agile methodologies, especially of Scrum. Major focus is applied to studies of scientific literature. In practical part of thesis is described authors' formulation of gamification and analysis of thematic studies. Then are evaluated results of semi-standardized interview with Scrum Master – on this basis is created appropriate gamification solution.

Keywords: Project management, Agile methodologies, Scrum, gamifying elements, Scrum framework, Scrum Poker, history of Agile methodologies, Waterfall, Agile manifesto, Scrum master

Obsah

Seznam obrázků

Seznam tabulek a grafů

Seznam matic

Seznam použitých zkratek

3M-SCRUM

3M	muri, mura, muda
Agile	z anglického agility (hbitost)
BDD	Behaviour driven development
CSM®	Certified ScrumMaster®
DEEP	detailed appropriately, estimated, emergent, prioritized
DSDM	Dynamic systems development method
FDD	Feature driven development
FSD	Federal Systems Division
IBM	International Business Machines
IID	Iterative and incremental development
IT	Information Technology
JIT	Just-in-time
Kanban	z japonského kanban (karta)
Lean	z anglického lean (štíhlý)
LEI	Lean Enterprise Institute
MDE	mechanics, dynamics, emotions
NASA	National Aeronautics and Space Administration
PaP	Pair programming
PBI	product backlog items
PDCA	plan, do, check, act
PDSA	plan, do, study, act
PoCo	points of consensus
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PT	skórovací body
RAD	Rapid application development
RPG	role-playing game
ROI	return of investment
Scrum	z anglického scrumage (skrumáž)

SCRUMBAN-XPE

Scrumban	kombinace Scrumu a Kanbanu
SDLC	Software development life cycle
SMART	specific, measurable, assignable, realistic, time-bound
SP	dovednostní body
TDD	Test driven development
UML	Unified Modelling Language
US#X	user story č. X
VSM	value-stream mapping
WiP	work in progress
XP	Extreme programming
XPE	zkušenostní body

Úvod

Agile – jedni v pojmu spatřují recept na úspěch, druzí něco málo registrují, třetí o něm neradi slyší atp. Protežovaný pojem se stal synonymem mnohého, ať už jednoduchosti, rychlosti, volnosti, nebo naopak chaosu, utopie, prázdnoty. Při současné vlně zájmu se nelze názorové rozpolcenosti divit. Turbulentní éra informací si žádá znalost i jiných přístupů než jen ryze tradičních, a tak hledání způsobů pro zdokonalení softwarového vývoje v organizaci, který reflektuje sílící propojenost informačních technologií s projektovým řízením, otevírá prostor pro různé úhly pohledu a hlubší diskuzi.

Mezi jednotlivými agilními metodikami výrazně převládá užití Scrumu. Určitou měrou tomu přispívá udělování příslušných certifikací, nicméně za pádnější argumenty jsou obecně považovány snadná aplikace, přímočarost či srozumitelnost. Zapomíná se ovšem na fakt, že se obtížně praktikuje. Procento organizací lpících na korektním rámci Scrumu není ve skutečnosti tak výrazné, jak na první pohled ukazují průzkumy. Existuje proto metodikou stanovená úloha Scrum mastera. Ten, velmi zjednodušeně, koriguje odchylky praktického Scrumu od doporučené teorie.

Jeden z nástrojů pro zlepšení si osvojení metodiky představuje gamifikace – běžně chápaná jako implementace herních prvků do neherního prostředí. Při vhodném provedení návrhu hry jde o motivační a vzdělávací aspekt, díky němuž dochází k pochopení a přijetí ucelených idejí Scrumu. Ačkoli se problematika gamifikace ve vztahu k softwarovému inženýrství teprve formuje, práce přispívá k podchycení jejího akademického potenciálu.

Cíl práce a metodika

Hlavním záměr práce tkví v návrhu ke zlepšení aplikace agilní metodiky Scrumu ve vybrané společnosti. Pro vytvoření důležitých teoretických poznatků je vymezena následující trojice bodů analytického charakteru:

1. Prostudování historie agilních metodik (viz kapitola 1).
   Historie agilních metodik se dá označit za vcelku probádané téma – naneštěstí málo ucelené. Je ho třeba znát z důvodu netradičního, přesto velmi přínosného rozměru chápaní Agilu, potažmo Scrumu.
2. Porovnání agilního přístupu s tradičním pojetím projektového řízení (viz kapitola 2).
   Detailní komparace stěžejních přístupů podmiňuje výzkumníka k podrobnější analýze problematiky. Není vhodné se zaměřit pouze na Agile bez rozvedení myšlenky Waterfallu, event. jejich kombinací.

3. Vymezení jednotlivých agilních metodik, především metodiky Scrum (viz kapitola 3).
   Agilní metodiky jsou mezi sebou do jisté míry provázané a mají mnoho společného. Všechny více či méně čerpají z principů Leanu. Mnohdy zahrnují pod sebou další metodiky nebo mohou být s nimi případně v přímé symbióze. Fokus na Scrum pak vytváří podklad pro návrh implementace gamifikace.

Praktická část se více zaměřuje na techniku gamifikace – od vlastního vysvětlení po návrh implementace gamifikačních prvků. Skládá se proto ze dvou bodů:

4. Definování gamifikace a rozbor praktického užití gamifikačních prvků (viz kapitola 1).
   Gamifikace nemá v odborné literatuře tak pevné ukotvení v návaznosti na aplikaci ve Scrumu. Další intence tedy směřuje k vlastnímu pojetí. To je podpořeno o studium souvisejících nezávislých studií.
5. Zpracování návrhu implementace gamifikace na projektu ve vybrané společnosti (viz kapitola 2).
   Druhý bod přímo koresponduje s cílem práce. Na základě příhodné kvalitativní metody v kombinaci se širokým teoretickým základem tvoří rozměr pro návrh gamifikace zlepšující aplikovaný Scrum.

Posledním dílčím záměrem je doporučení dalšího postupu/výzkumu, a to kvůli žádoucímu podnětu k dalšímu podrobnému řešení problematiky odborníky a zájemci.

Metodické zaměření práce se zakládá na podrobném studiu dokumentů a pramenů objasňujících relevantní výklady pojmů. Cennými zdroji jsou proto jak základní dokumenty vymezující rámce jednotlivých přístupů, metodik a formulací, tak časopisecké vědecké publikace pojednávající o zkoumaných tématech a příslušných studiích.

Za nezbytnou součást praktické části platí zveřejnění výsledků uplatnění kvalitativní metody polostandardizovaného rozhovoru s odborníkem z praxe – certifikovaným Scrum masterem organizací ScrumAlliance.

Oba vědecké přístupy jsou východiskem pro stanovení návrhu implementace gamifikace v konkrétní společnosti. Podle výstupů rozhovoru se identifikují konkrétní nedostatky v užívaném Scrumu a v návrhu nakonec zakomponují nerozvinuté, potlačované nebo chybějící elementy uplatňované metodiky.

Teoretická východiska

Komplexní podstata práce spočívá ve vytvoření výchozího teoretického základu pro pozdější srovnání s praktickým užitím ve vybrané společnosti – pro to poslouží zejména závěrečná praktická část.

První kapitola práce nejprve objasňuje historii agilních metodik. Od té přechází ke srovnání Waterfall x Agile. Pak definuje jednotlivé agilní metodiky. Protože práce řeší konkrétně Scrum, věnuje prostor zvláště jeho rozboru.

1 Historický vývoj agilních metodik

Vzhledem k tomu, že Agile reprezentuje jeden z řady přístupů, jak je možné přistoupit k vývoji softwaru, základním východiskem pro podobu vůbec všech modelů či paradigmat byly praktické poznatky našich předchůdců. V dávné minulosti používali metodu pokus-omyl nebo nabývali zkušenosti prostřednictvím učení se od mistrů svého oboru, popř. děděním rodinných znalostí.

Často se proto nedá vytvořit konkrétní přehled o tom, kdy, kde a kým byly jaký model, potažmo metoda, původně použity, protože takové údaje mohou být bez pádných důkazů i v moderních dějinách vágní nebo zpochybnitelné. Mimoto se softwarové inženýrství nachází v ustálené podobě zhruba půl století.

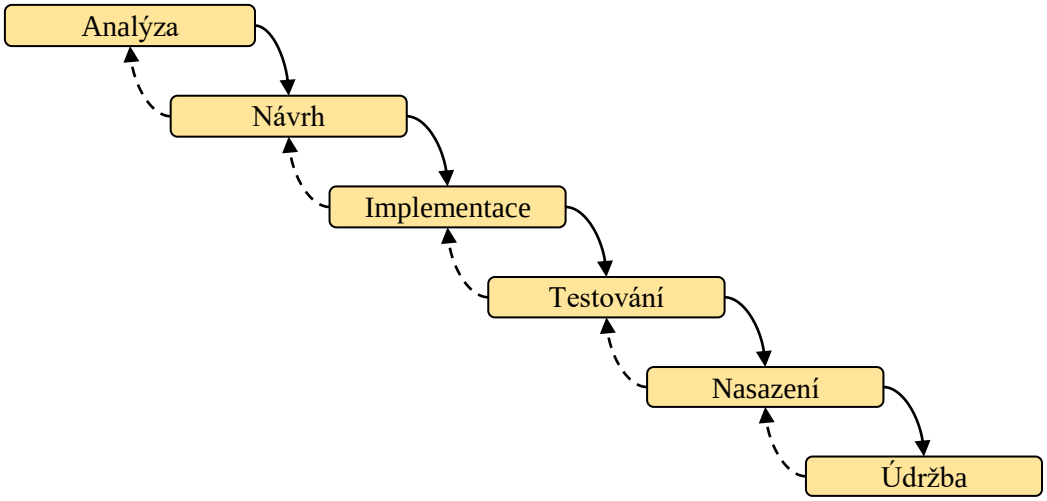
Podkapitola níže rozebírá historicky nejvýraznější impulz pro aplikaci Agilu. Slouží jako výchozí bod pro chronologické zachycení podkladově prokazatelného vývoje agilních metod zpětně od 30. let minulého století až do začátku nového milénia.

1.1 Agile jako reakce na slabé stránky Waterfallu

Většina odborných studií a výzkumů uvádí Agile jako odpověď na otázku nedostatků klasických přístupů. Slovy Cohena se jedná o "reakci na tradiční způsoby vývoje softwaru" (Cohen, a další, 2004). Günal určil stejnou příčinu, načež konkretizuje, že jsou v případě Waterfallu problémové "vysoce dokumentově orientované způsoby vývoje a takové modely, které striktně omezují vývojáře a vyžadují od nich, aby uplatňovali stanovené procesy" (Günal, 2002).

Obrázek č. 1 znázorňuje model životního cyklu vývoje softwaru demonstrující princip Vodopádu a charakterizující pevnou návaznost aktivit včetně zpětných vazeb první úrovně:

Obrázek č. 1 – Model Waterfallu (Vodopádu)



Zdroj: vlastní zpracování

Kaskádová varianta je ve srovnání s agilitou interpretována ve druhé kapitole této práce. V historickém kontextu postačí zatím přesněji formulovat slabiny Waterfallu. Petersen poukazuje na vysoké náklady a nadbytečné úsilí (Petersen, a další, 2009), což způsobuje:

- notné množství dokumentů – administrativní zátěž způsobuje nepřehlednost a snižuje srozumitelnost,
- obtížné provádění změn – lineární model disponuje ze své podstaty velmi omezenou dynamičností a
- dodání výstupu na konci projektu – produkt je k dispozici až na konci řešení.

Kritika tradičního přístupu, jak bylo zmíněno, existovala již ve 20. století. Tehdejší alternativní myšlenky akorát nebyly definovány jako agilní. Podle Abbase „bohužel nebyly brány dostatečně vážně, takže trvalo dalších 30 let, než se přišlo na to, že jsou efektivním způsobem, jak vyvíjet software“ (Abbas, a další, 2008).

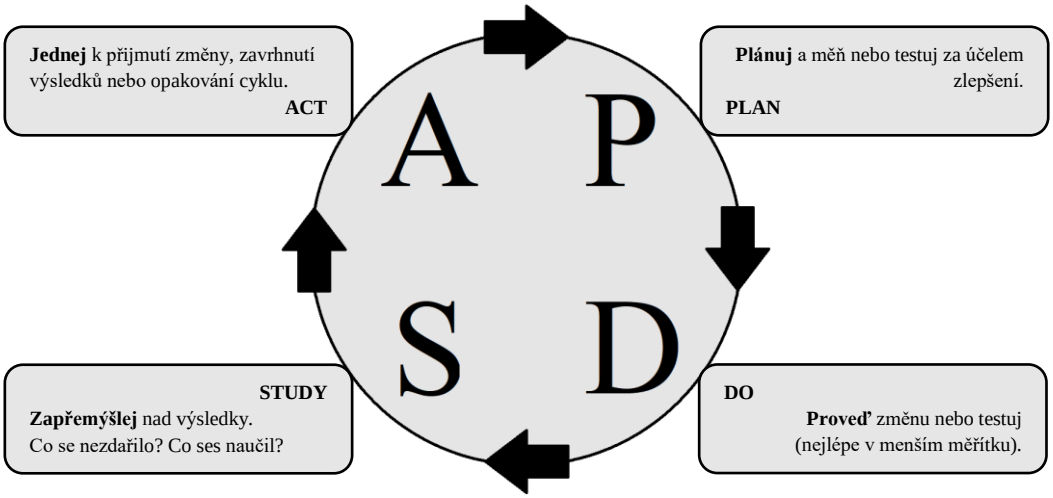
1.2 Agilní metodiky ve 20. století

V této podkapitole je třeba brát zřetel na skutečnost, že agilní metody existovaly mnohdy dříve. Smysl tím pádem tkví v odhalení momentů formalizace metod či jejich nedílných součástí, případně jiných milníků.

1.2.1 PDSA jako základ IID (30. – 40. léta)

Počátky iterativního a přírůstkového vývoje (IID – Iterative and incremental development) jakožto předvoje Agilu se objevují ve 30. letech v podobě návrhu techniky PDSA (nepřesně PDCA), známé také jako Shewhartův cyklus (Shewhart, 1939). Podstata spočívá v provádění iterací cyklu do té doby, dokud nedojde k vyřešení konkrétního problému – zlepšení kvality. Ke správné posloupnosti procesů navádí samotná zkratka PDSA, která je konkrétněji vysvětlena zde:

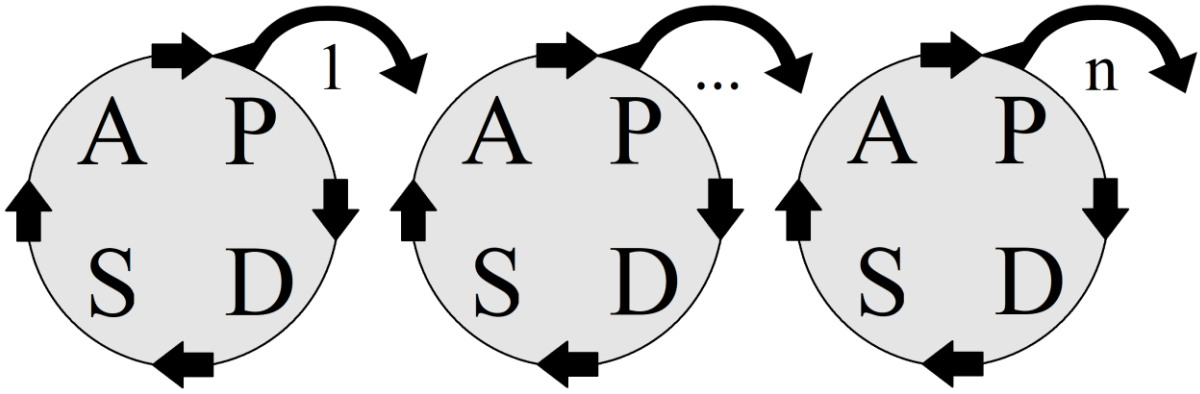
Obrázek č. 2 – Shewhartův cyklus učení a zlepšování procesu/produktu



Zdroj: Deming, 1993

Více cyklů PDSA může být zahájeno buď v řadě, nebo simultánně. Obrázek č. 3 poukazuje na reálně užívanější princip sekvence:

Obrázek č. 3 – Sekvenční model PDSA cyklů



Zdroj: vlastní zpracování

Aktivním propagátorem PDSA byl po zbytek století „otec řízení kvality“ W. Edwards Deming. Vyučoval o něm už od druhé světové války (především pak v povalečném Japonsku), a to se záměrem napomáhat vytvářet nové organizační schopnosti (Best & Neuhauser, 2005).

40. léta znamenala mimo jiné počátek hlubších studií japonské automobilové společnosti Toyota, na jejichž základě Ohno později definoval ve své knize Lean a Kanban (Ohno, 1988). Předmětem zájmu byly praktiky tehdejších supermarketů, díky nimž bylo možné získat náhled na problematiku právě včasné (JIT – just-in-time) produkce (viz str. 51).

1.2.2 IID a XP v leteckém a kosmickém průmyslu (50. – 60. léta)

Dalším patrným krokem byla aplikace IID v 50. letech. Vedla k úspěšnému vytvoření experimentálního raketového letounu North American X-15 pro NASA. Ačkoli se nejednalo o softwarový projekt (Dana, 2004), dokázal rozšířit znalostní bázi svých pracovníků. Ti napomohli na přelomu 60. let k užití IID v softwarové části Mercury, kde bylo využito také extrémního programování (XP – Extreme programming). Hlavním cílem projektu sice bylo dosáhnout řízeného orbitálního letu s bezpečným návratem, ale pod tíhou studené války NASA vyžadovala také vyvinutí nových technologií (Brownlee Jr., 2009).

Ke konci 50. let se ke kořenům IID hlásí také společnost IBM, jejíž divize federálních systémů (FSD – Federal Systems Division) se také podílela na projektu Mercury (Larman & Basili, 2003). Příkladů aplikace IID s participací FSD se v projektech NASA dá v pozdějších letech nalézt více (např. u projektu Space Shuttle v 70. letech)

60. léta zároveň provázela softwarová krize, kdy bylo běžné markantní prodlužování a prodražování projektů (Fitzgerald, 2012). Stala se podnětem k rozvoji softwarového inženýrství v těchto a následujících letech.

1.2.3 Rozvoj softwarového inženýrství (70. – 80. léta)

70. léta jsou příznačná pro počátky kaskádového způsobu řízení. Roycův článek „Managing the Development of Large Software Systems“ přinesl myšlenku tvořící základ Waterfallu (Royce, 1970). Larman s Basilim v něm objevili náznaky IID, zpětné vazby či adaptace (Larman & Basili, 2003).

Tradiční přístup byl postupně více uplatňován, čímž došlo k patrnému odhalení jeho slabých stránek. Ganis objasňuje důvody stupňující se nespokojenosti, kdy „na konci 70. let

začaly vznikat problémy s formalitou a sekvenční povahou vodopádových procesů“ (Ganis, 2010). Objekt kritiky tudíž zůstává od svého počátku identický, třebaže Agile neznamená historicky jedinou snahu o řešení.

Během poloviny 80. let byly v reakci na omezení Waterfallu rozvinuty nové modely, nejvýrazněji pak model Rapidního prototypování nebo model Spirály. Oba vychází z konceptu softwarového prototypu, jehož účel vyplývá z nahrazení písemných požadavků Waterfallu nefunkční pracovní verzí konečného produktu, a to kvůli získání zpětné vazby od zákazníka ke zlepšení designu systému v raných fázích vývoje. Zatímco takový draft sloužil zejména pro demonstraci, ve fázi návrhu a implementace byl opět užíván kaskádový model, což nežádoucně vytvořilo závislost prototypu na Waterfallu včetně jeho nedostatků (Monochristou, a další, 2005).

Ačkoli původní účel těchto modelů selhal, povedlo se tím položit základ metody Rapidní aplikace vývoje (RAD – Rapid application develepment). Byť byla užívána už v druhé polovině 80. let, Martin o ní pojednává v „Rapid Application Development“ až roku 1991 (Martin, 1991).

V případě agilních metod byla velmi významná již naznačená Ohnova publikace „Toyota seisan hoshiki – Datsu-kibo no keizai o mezashite“ z roku 1978 (do angličtiny přeložena o deset let později), která popisuje Lean (potažmo „štíhlou výrobu“ – Lean manufacturing) – základ mnoha dalších metod a Kanban jako optimalizační způsoby k urychlení procesu mezi objednávkou a platbou zákazníka (Ohno, 1988).

Roku 1986 byl vymezen Scrum. „Otcové Scrumu“ Takeuchi a Nonaka totiž sepsali základní dokument této metodiky pod názvem „The New New Product Development Game“ (Takeuchi & Nonaka, 1986). Podařilo se tak sestavit komplexní teorii adaptivních systémů na bázi Leanu uplatňovaných právě v Toyotě nebo Hondě ve spojení se strategiemi znalostního managementu autorů výše uvedené práce.

Nakonec stojí za zmínku aplikace IID s participací IBM FSD u hlavního softwaru v projektu Space Shuttle. Ten měl za cíl uskutečňování pilotovaných letů s pomocí raketoplánů. V souvislosti s uplatněním IID se jedná o konkrétní období 1977–1980, kdy bylo názorně použito 17 iterací po dobu 31 měsíců (Madden & Kyle, 1984).

1.2.4 Agile movement (90. léta – uvedení Manifestu)

Za výchozí podnět pro rozvoj metodologií vývoje softwaru ke konci 20. století, souhrnně označovaném jako Hnutí Agilu, se považuje rozvoj Krystalových metod (Crystal methods – zkráceně Crystal). Vychází z Cockburnových letitých studií a rozhovorů v rámci týmů. Výsledek jeho snažení byl zveřejněn v roce 2004 jako „Crystal clear a human-powered methodology for small teams“ (Cockburn, 2004). Cockburn je jedním ze signatářů Manifestu pro agilní softwarový vývoj (dále jen Manifest, viz další kapitola).

Navazující metodou RAD se stala Metoda vývoje pomocí dynamických systémů (DSDM – Dynamic systems development method). Oficiálně byla založena roku 1994 Konsorciem DSDM (DSDM Consortium) sdružujícím vendory a odborníky z oblasti softwarového inženýrství. DSDM vznikla za účelem rozvoje a podpory nezávislého rámce RAD v kombinaci s nejlepšími zkušenostmi praktiků. Fowler, také jeden z autorů Manifestu, tvrdí, že „DSDM zasluhuje pozornost díky tomu, že obsahuje více infrastruktury zralejších tradičních metod v souladu s principy přístupu agilních metod“ (Fowler, 2005). Stěžejní dokument této metody zhotovila na konci dekády Stapleton pod názvem „DSDM: Dynamic Systems Development Method“ (Stapleton, 1999).

Třebaže se Scrum objevil již v 80. letech, rozsáhlejší deskripce se dočkal roku 1995 v podobě „SCRUM Development Process“ od Schwabera (Schwaber, 1995), dalším tvůrcem Manifestu.

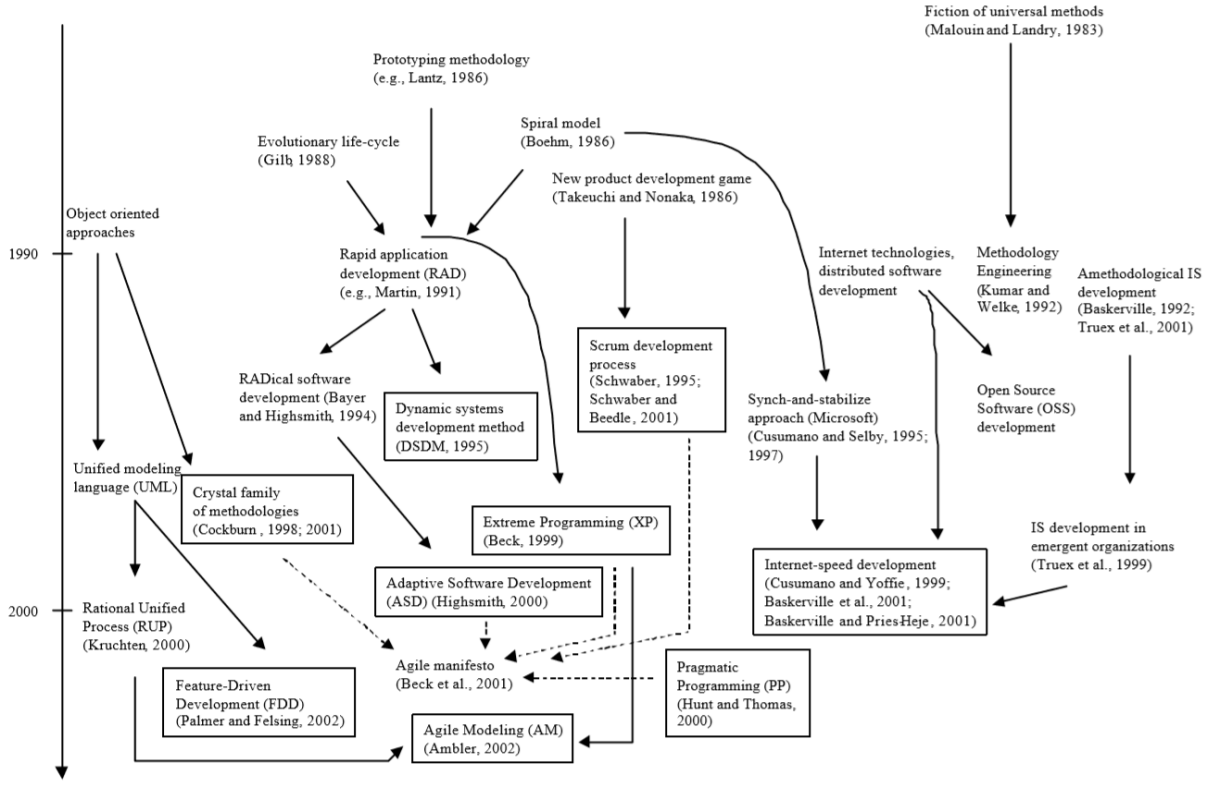
Další metodika, vývoj řízený užitnými vlastnostmi (FDD – Feature driven development), byla zformována na přelomu století. De Luca navázal na své předchozí zkušenosti z IBM laboratoří pro programování (programming laboratories) a využil je ve spolupráci s Coadem v letech 1997–1998 na projektu mezinárodní banky se sídlem v Singapuru právě v podobě FDD (Roock, 2007).

Že nejsou Coad či De Luca označováni za „otce FDD“ nadarmo, dokazuje kniha „Java modeling in color with UML: enterprise components and process“ (Coad, de Luca, & Lefebvre, 1999). Konkrétnějším pramenem k metodě je pak „A Practical Guide to Feature-Driven Development“, publikovaný již v novém tisíciletí (Palmer & Felsing, 2002).

Další jméno spjaté s Manifestem, Beck, symbolizuje staronovou formalizovanou metodu, tentokrát zachycenou v knize „Extreme programming explained: embrace change“ (Beck, 1999). Metoda XP kloubí dohromady osvědčené postupy pro plánování, řízení, návrh, kódování a testing. Beck jako součást XP představil též koncept user stories (viz str. 40).

Protože rozmach výše uvedených a dalších agilních metodik proběhl v relativně nedávné době, lze hnutí Agilu zobrazit na krátké časové ose dle roku vydání jejich významných publikací, viz obrázek č. 4 (plné šipky znamenají přímou návaznost, čárkované částečnou):

Obrázek č. 4 – Evoluční mapa hnutí Agilu



Zdroj: Abrahamsson, 2003

1.3 Manifesto for Agile Software Development

Vytvoření Manifestu na začátku 21. století zapříčinilo na poli softwarového inženýrství zásadní změny. Byl publikován v roce 2001 mezinárodní skupinou praktiků a konzultantů (Beck, a další, 2001) jakožto dovršení Hnutí Agilu.

Během třídenního únorového setkání v utahském lyžařském rezortu The Lodge at Snowbird v pohoří Wasatch panovala nálada nastavit alternativu k dokumentově orientovaným a těžkým (heavyweight) procesům softwarového vývoje (Highsmith, 2001). Od té doby se pojednává o agilních metodách současně jako o lehkých (lightweight) metodách.

K hlubšímu porozumění Manifestu a jeho čtyř hodnot či dvanácti principů nestačí uvádět pouze jejich heslovitá znění – jednoduše proto, že v sobě zahrnují pokrokové a komplexnější myšlenky. Věnují se jim následující podkapitoly.

1.3.1 Čtyři hodnoty softwarového vývoje

Jako základní hodnoty prezentuje:

- 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

Každé z hesel je podrobněji rozebráno v dalších řádcích (Abrahamsson, a další, 2002):

JEDNOTLIVCI A INTERAKCE PŘED PROCESY A NÁSTROJI

Daný princip poukazuje na roli člověka, tj. vztah a komunitu developerů v kontrastu s vyhrazenými procesy a vývojovými nástroji. Pozornost je zaměřena na interpersonální vztahy v týmu a na podpůrné pracovní prostředí. Jedná se tedy o zdůraznění týmového ducha.

FUNGUJÍCÍ SOFTWARE PŘED VYČERPÁVAJÍCÍ DOKUMENTACÍ

Jeden z elementárních cílů týmu vývojářů spočívá v neustálém testování softwaru. Nové verze vznikají v pravidelných intervalech, např. na měsíční bázi. Co se týče kódu, mají vývojáři za úkol zajistit jednoduchost, srozumitelnost a co největší možný pokrok. Dochází tak k redukci nadbytečné dokumentace na optimální úroveň.

SPOLUPRÁCE SE ZÁKAZNÍKEM PŘED VYJEDNÁVÁNÍM O SMLOUVĚ

Projev upřednostňuje vztah mezi vývojáři a klienty nad přísností smluv. Nutno podotknout, že u dobře sestavené smlouvy roste důležitost stejně jako velikost softwarového projektu. Proces jednání by měl být sám o sobě chápán jako prostředek k dosažení a udržení životaschopného vztahu. Agile je z hlediska byznysu zaměřen na přínos obchodní hodnoty hned, jak projekt začne, čímž se snaží vyvarovat riziku neplnění smlouvy.

REAGOVÁNÍ NA ZMĚNY PŘED DODRŽOVÁNÍM PLÁNU

Další kategorie, tzv. vývojová skupina, zahrnující jak vývojáře softwaru, tak představitele klientů, by měla být dobře informována, oprávněna a autorizována, aby se zvážily eventuální úpravy, které během životního cyklu vývoje nastávají. Účastníci jsou tedy připraveni provádět změny za předpokladu existence smluv umožňujících a podporujících proces vylepšování.

1.3.2 Dvanáct principů softwarového vývoje

Autoři se v dokumentu dále řídí těmito principy (Beck, a další, 2001):

- Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
- Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
- Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
- Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
- Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
- Nejúčinnějším a nejefektnějším způsobem sdělování informací ve vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
- Hlavním měřítkem pokroku je fungující software.
- Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
- Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
- Jednoduchost – umění maximalizovat množství nevykonané práce – je klíčová.
- Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
- Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

Předešlá podoba principů je výsledkem formulace dle českého překladu vnořeného v internetovém zdroji, kde se Manifest přímo nachází. Přestože je popisnější než v případě čtyř hodnot, nabízí se pro srovnání interpretace opřená o Liho postřehy (Li, 2012):

SPOKOJENOST ZÁKAZNÍKŮ PROSTŘEDNICTVÍM ČASNÉHO A PRŮBĚŽNÉHO POSKYTOVÁNÍ SOFTWARU

Princip má zjevnou vazbu na metodu IID. Staví totiž na průběžném doručování výstupů (iterativní proces) a doplňuje ho o včasnou dodávku softwaru (inkrementální složku). Snahou je porozumět požadavkům zákazníka a zároveň vytvořit zpětnou vazbu pro vývojáře.

Očekávané přizpůsobování se měnícím se požadavkům během celého procesu vývoje

Celým procesem se myslí také pozdní vývoj, během kterého je stále možné provádět změny podle příslušných požadavků. Jestliže software splňuje potřeby uživatele a trhu, prioritou se posléze stává software jako takový. Brooks míní, že „nejtěžší částí tvorby softwarového systému je rozhodnout, co budovat," a že „klient obvykle neví, na jaké otázky je třeba zodpovědět, a téměř nikdy nepřemýšlí nad problémy v detailech potřebných pro specifikaci" (Brooks, 1986). Díky tomu, že agilní metody se změnami požadavků počítají, dovedou minimalizovat náklady z nich vyplývající.

Krátká pravidelná dodávka funkčního softwaru

Zde jsou elementární iterace v krátké době (dny, týdny, měsíce) zajišťující projektovému týmu užší spolupráci s uživatelem. Projektový tým při každé nové dodávce poskytuje vylepšení nebo funkce na základě předešlé dodávky. Tyto nové prvky musí být nejprve otestovány, aby poté uměly správně pracovat a na závěr odpovídat požadovanou kvalitou.

Spolupráce mezi lidmi z byznysu a vývojáři v průběhu celého projektu

V momentě, kdy zástupci byznysu přetlumočí požadavky klienta vývojářům, není výjimkou, že jim leckdy neporozumí. Myšlenka proto směřuje ke kooperaci mezi vývojem a byznysem za účelem smysluplné a včasné reakce, když se problémy obdobného rázu vyskytnou. Cockburn vysvětluje: „Čím déle trvá přenos informace od a k vývojářům, tím více škody to napáchá na projektu" (Cockburn, 2006).

Řízení projektu v okolí motivovaných jedinců

Individuím je třeba dopřát patřičnou podporu a prokázat vkládanou důvěru. Daří-li se princip uplatnit, tým pak dokáže disponovat plným potenciálem.

Interakce tváří v tvář

Za nejúčinnější způsob předávání informací v rámci vývojového týmu se považuje přímá komunikace. Agile tým se skládá z pár členů, a proto je vhodné konverzovat tváří v tvář. Napomáhá tomu práce ve společném prostředí a konání schůzek. Ačkoli komunikace face-to-face přináší mnoho výhod (redukuje administrativní zátěž, umožňuje flexibilnější

přenos informací atd.), nemůže být nutně jediným způsobem, jak získávat, sdílet a ukládat informace.

FUNKČNÍ SOFTWARE JAKO PRIMÁRNÍ UKAZATEL POKROKU

Zásadou každé iterace je dodání funkčního softwaru. V tom případě ukazatel progresu nezastupuje množství řádků kódu nebo implementovaných testovacích případů, nýbrž počet otestovaných softwarů, které splňují standardy pro uvedení a jsou provozuschopné.

Princip se tudíž vztahuje na výstup vývojového procesu v rámci metrik produktivity. Mimoto lze určit metriky:

- kvality – týkají se principu o spokojenosti klientů a o připravenosti se adaptovat na změny,
- technického rázu – souvisí s principem o koncentraci na technický detail a o týmech s vlastní organizací,
- zaměřené na člověka – jsou vázány na princip o pravidelných úvahách nad vlastní efektivitou a
- orientované na velikost – jsou do jisté míry zahrnuty také v principu o soustředěnosti na technický detail.

AGILNÍ PROCESY PODPORUJÍCÍ KONZISTENTNÍ TEMPO VÝVOJE

Unavený a stresovaný pracovník neodpovídá myšlence Agilu. Aspekt sociální odpovědnosti vybízí k respektování úměrné pracovní doby, což v kontextu přetěžování zaměstnanců směřuje k jejich větší efektivitě a dosažení lepších výsledků v rámci projektu.

SOUSTŘEDĚNOST NA TECHNICKÝ DETAIL A DESIGN ZLEPŠUJÍCÍ AGILITU

Refaktoring (zdokonalování a čištění kódu) sice představuje posílení agility, ale jen do jisté míry. Při nepotřebném refaktoringu dochází z pohledu projektového manažera ke zbytečné ztrátě zdrojů. To samé platí v případě designu.

JEDNODUCHOST

Ačkoli je v programování složitější vytvořit jednoduchý design než komplexní řešení, zabývat se budoucími funkcemi softwaru namísto aktuálních požadavků není očekávaným přístupem agilního týmu. Vývojáři by měli zajistit akorát funkce, které byly explicitně dohodnuty se zákazníkem.

TÝMY S VLASTNÍ ORGANIZACÍ NAPOMÁHAJÍCÍ ZDAŘILÝM ARCHITEKTURÁM, POŽADAVKŮM A DESIGNŮM

Samo-organizující se týmy mají schopnost komunikovat mezi sebou tak, že tím zároveň formují společnou firemní etiku a kulturu. Mezi členy týmu pak panuje větší důvěra a klesá potřeba podrobných instrukcí.

PRAVIDELNÉ ÚVAHY O ZLEPŠENÍ EFEKTIVITY

Vývojáři by se měli pravidelně setkávat i proto, aby vzájemně debatovali o svých pracovních návycích a užívaných metodách. Ty je zapotřebí přizpůsobit právě agilnímu smýšlení.

2 Srovnání tradičního projektového řízení (Waterfall) oproti agilnímu řízení (Agile)

Ať už je předmětem diskurzu Waterfall nebo Agile, opravdu představují jediné reprezentanty životního cyklu softwarového vývoje (tzv. SDLC – Software development life cycle)? Nikoli, protože se na stejnou úroveň řadí okrajově zmíněný model Spirály (viz str. 7), V model a další modely či paradigmata – ty nejsou ovšem předmětem práce.

V první řádě je rozebrán model Vodopádu, který reprezentuje tradiční přístup a je příznačný těžkými procesy softwarového vývoje. Následuje charakteristika představitele lehkých metod, Agilu coby paradigmatu. Srovnáním jako takovým mezi pojetími se zabývá třetí část této kapitoly. Poslední podkapitola otevírá náhled do problematiky Hybridu.

2.1 Waterfall

Waterfall zapadá do konceptu tradičního projektového řízení. Adenowové tvrdí, že "model Waterfallu je statický model, který přistupuje k vývoji systémů lineárním a sekvenčním způsobem, tj. po dokončení jedné fáze nastává druhá" (Adenowo & Adenowo, 2013).

V první kapitole byly kvůli historickému kontextu zmíněny výhradně negativní stránky Waterfallu (viz str. 4). Má však i své nesporné klady, mezi které podle Parekha patří (Parekh, 2018):

- ✓ existence jasného rozdělení práce a kontroly. Díky tomu je snazší nastavit plán úkolů, které mají být dokončeny ve vymezené době.
- ✓ pevné stanovení prací pro určitou fázi. Nedochází tak k jejich překryvům a potenciálně nadbytečným iteracím.
- ✓ snazší předvídatelnost. Protože procesy probíhají lineárně, snižují se do značné míry náklady na zdroje (tím i projektu).
- ✓ dokumentace a testování na konci každé fáze. Zlepšuje se tak kvalita projektu.

Pozornost upoutá, že jmenovitě poslední plus je z druhé strany slabinou Waterfallu a de facto důvodem, proč vznikl Manifest.

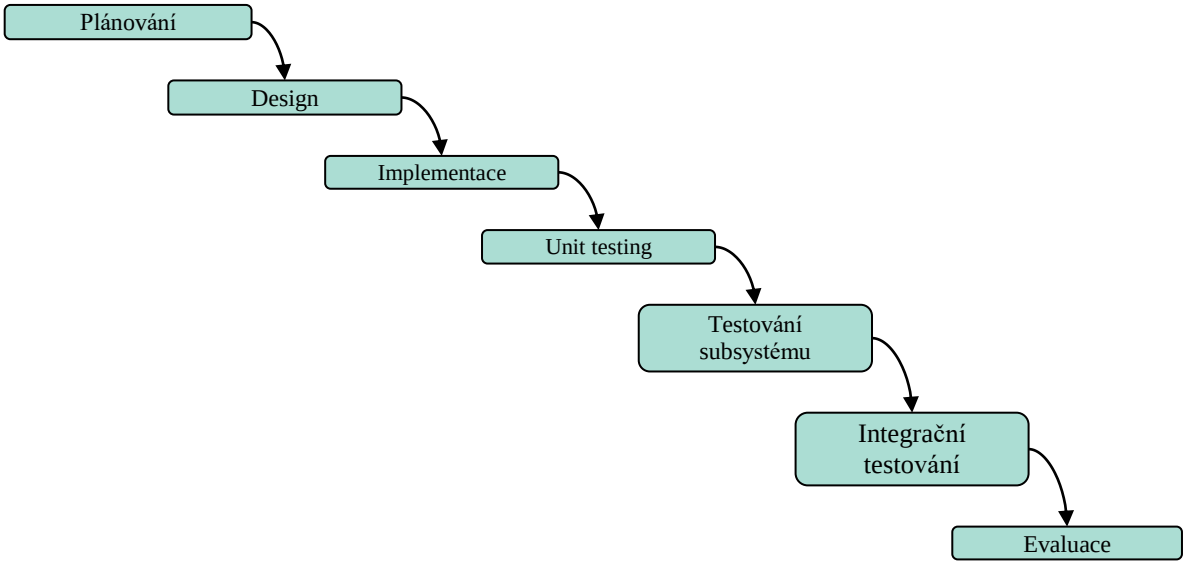
Následující odstavec poukazuje na kardinální rozdíl mezi Waterfallem a jeho předchůdcem, kdy nešťastné přirovnání k vodopádu evokuje rozdílný model. Fáze zůstávají

víceméně identické, proto jsou blíže vysvětleny v odstavci 2.1.2. Za nejznámější modifikaci Waterfallu pak platí Sashimi model (odstavec 2.1.3).

2.1.1 Srovnání se Stagewise modelem

V předchozí kapitole byl zmíněn Roycův článek ze 70. let (viz str. 6), považovaný za průkopnický ve vztahu k Waterfallu, nicméně už v roce 1956 vyšla Beningtonova publikace „Production of Large Computer Programs“, v níž autor rozvedl striktní posloupnost jednotlivých fází, tzv. Stagewise model (Benington, 1956):

Obrázek č. 5 – Stagewise (stupňovitý) model



Zdroj: vlastní zpracování

Takový model disponuje řadou nevýhod, protože je příliš přímočarý a nebere v potaz identifikaci možných rizik anebo vůbec jakékoli revize. Royce právě rozpoznal, že po ukončení jednotlivých fází mohou nastat potíže s konstrukcí (Royce, 1970). K původnímu modelu přidal návaznou zpětnou vazbu jako v obrázku č. 1. I když Royce ve svém dokumentu nikdy nepoužil pojem Waterfall, Beningtonův model rozšířený o jeho iterativní zpětnou vazbu je tak v současnosti nazýván. Někteří autoři přesto chybně zaměňují Stagewise model za Waterfall (např. Manzoor, 2015), nebo ho rozlišují na klasický Waterfall a Waterfall s iteracemi (např. Subair, 2014), což svádí k domněnce, že je feedback pro Waterfall volitelným instrumentem, byť je jeho esencí.

Dojde-li k porovnání mezi obrázky č. 1 a 4, jednotlivé kroky a jejich počet se liší. Není to však z důvodu nové struktury fází modelu, nýbrž díky modifikacím v různých interpretacích, se kterými je třeba v literatuře počítat (kupříkladu v obrázku č. 4 je kladen

důraz na rozpad částí testování). Ani Beningtonův model se v původním znění neshoduje s Roycovými nákresy, přesto oba modely reflektují stejnou myšlenku životního cyklu. Jinými slovy, pokud by byla Roycova iterativní zpětná vazba v Stagewise modelu na předchozí straně doplněna, vznikne prakticky tentýž model Waterfallu jako na příkladu obrázku č. 1.

2.1.2 Fáze Waterfallu

Jak je zřejmé, výklady nejsou zcela unifikovány v názvosloví nebo členění fází. Všeobecnější variantou se jeví členění v obrázku č. 1. V rozpadu je z něj dále vycházeno a jednotlivé fáze chronologicky interpretovány:

ANALÝZA

V podstatě se jedná o vymezení softwarových požadavků – kompletní a podrobný popis chování zamýšleného softwaru. Jak systémoví, tak business analytici definují funkční a nefunkční požadavky. Stephens s Rosenbergem odkazují na DeGraceova kritéria (DeGrace & Stahl, 1993), „jež se často označují jako „osti“: schopnost, udržovatelnost, použitelnost, znovupoužitelnost, spolehlivost a rozšiřitelnost“ (Stephens & Rosenberg, 2017). Funkční požadavky jsou vázany na schopnost, nefunkční potom na všechny zbylé znaky.

Pomocí případů užití (use cases), které napomáhají zachytit a popsat interakce uživatelů se softwarem, jsou formulovány první z požadavků. Obsahují požadavky jako účel, perspektivu, rozsah (scope), funkcionalitu (rozsah funkcí), funkce, atributy, uživatelské vlastnosti, specifikace, požadavky na rozhraní atd. Nefunkční požadavky se týkají zase různých omezení a požadavků spíše k návrhu a provozu softwaru nežli ke konkrétnímu chování.

NÁVRH

Designová část navazuje plánováním a řešením problému softwarového řešení. Od návrhářů a vývojářů softwaru se očekává zpracování plánu řešení v podobě návrhů algoritmu, architektury softwaru, grafického uživatelského rozhraní, logického diagramu, schématu databázové koncepce, aj.

Dle Kassema mezi podstatné výstupy fáze návrhu patří tzv. high-level návrh a podrobná návrhová dokumentace. Architektonický high-level návrh cílí na identifikaci softwarových modulů a jejich rozhraní. Kromě toho jsou v něm vyjádřeny dle autora „záležitosti ohledně robustnosti, škálovatelnosti, bezpečnosti, odolnosti proti chybám a

testovatelnosti“. Podrobná návrhová dokumentace dává k dispozici podrobnosti o každém jednotlivém modulu vyjádřeném v high-level návrhu. Detaily zahrnují struktury dat a algoritmy pro implementaci jednotlivých modulů (Kassem, 2009).

IMPLEMENTACE

Zjednodušeně lze hovořit o procesu přeměny všech plánů a požadavků na produkční prostředí. Během ní se realizují požadavky do podoby spustitelného programu, webové stránky, databáze nebo jiného softwarového komponentu. V této fázi se zejména zavádí a programuje. Je zhotoven skutečný kód a kompilován do operační aplikace, proto může být fáze označována taktéž jako kódovací. Stejně jsou vytvořeny příslušné textové soubory a databáze.

Podle Boswortha „mnoho vývojářů považuje za velký problém strukturu modelu, ve kterém dochází k implementaci dřív, než je systém skutečně připraven k předání uživateli,“ ačkoli zastánci Waterfallu tvrdí, že konzistentní testování v průběhu celého vývojového procesu umožňuje připravenost systému již v momentě, kdy je této fáze regulérně dosaženo (Bosworth & Kabay, 2002).

TESTOVÁNÍ

V této fázi se v rámci systémové integrace testuje kvalita společně s funkčními aspekty. Vzhledem k tomu, že se očekává kompletní řešení (hardwarové i softwarové), musí být testy prováděny v různých konfiguracích a také samotným klientem. Výsledek testingu je ověřen přes kontrolní seznam, tj. zda byl systém verifikován (verifikace) a vyskytují-li se odchylky z hlediska kvality a času oproti předchozím rozhodnutím ke kontrole kvality (ladění a validace), zda jsou definovány plány na předání produktu zákazníkovi podle podnikových směrnic a zda výsledek projektu splňuje jeho požadavky.

Ve vztahu ke kódu se snaží opravit jeho chyby a zahrnuje integraci. Bitter podotýká, že: „ne vždy se klade testování taková důležitost. Nezáleží na stráveném čase, ale na nalezení všech chyb, protože to je to, co je značně složitější“ (Bitter, Mohiuddin, & Nawrocki, 2000).

NASAZENÍ

Nasazení (do produkce) představuje část Waterfallu, kdy je nový software představen a připraven na aktuální užívání zákazníkem. Xiong rozvádí, že společně s podporou (supportem) na pomyslné hranici mezi nasazením a údržbou v sobě obsahuje procesy

„zveřejnění (release), instalace, adaptace, rekonfigurace, aktualizace, aktivace, deaktivace, odinstalace a úplného vyřazení," (Xiong, 2011). Do výčtu by se dala zahrnout i migrace dat. Často je zmiňována výlučně instalace, např. když ji Garrido označuje za „výslednici dodávky (delivery)," (Garrido, 2013), ale rozhodně není synonymem pro nasazení jako takové.

ÚDRŽBA

Předchozí etapa není považována za konečnou, protože je často nutné pokračovat v údržbě uvedeného softwaru, aby mohl operovat na optimálních úrovních výkonu. Řešit se mohou bezpečnostní otázky, dříve neobjevené chyby, problémy jiného charakteru, které v rámci implementace či pokračujících stadiích neměly prioritu v řešení atd. Laplante označuje tyto aktivity za reinženýringové (ve smyslu inovativní) a třídí je na (Laplante, 2007):

- adaptivní – jsou výsledkem vnějších změn, na které systém musí reagovat (používají se k vylepšení systému přidáním funkcionalit a funkcí v návaznosti na technologický pokrok)
- korektivní – vyžadují údržbu k opravě chyb (týkají se odstraňování chyb a bugů ze systému)
- perfektivní – reprezentují další údržbu (třeba úpravu dokumentace, vylepšení funkcí s ohledem na dodatečné požadavky uživatele, zlepšení kvality a efektivity aj.)
- preventivní – autor je nezmiňuje pravděpodobně proto, že mají perfektivní charakter, ale svoji pozornost si zasluhují (směřují totiž k předvídání možných problémů a zajištění jejich nápravy předtím, než k nim vůbec dojde)

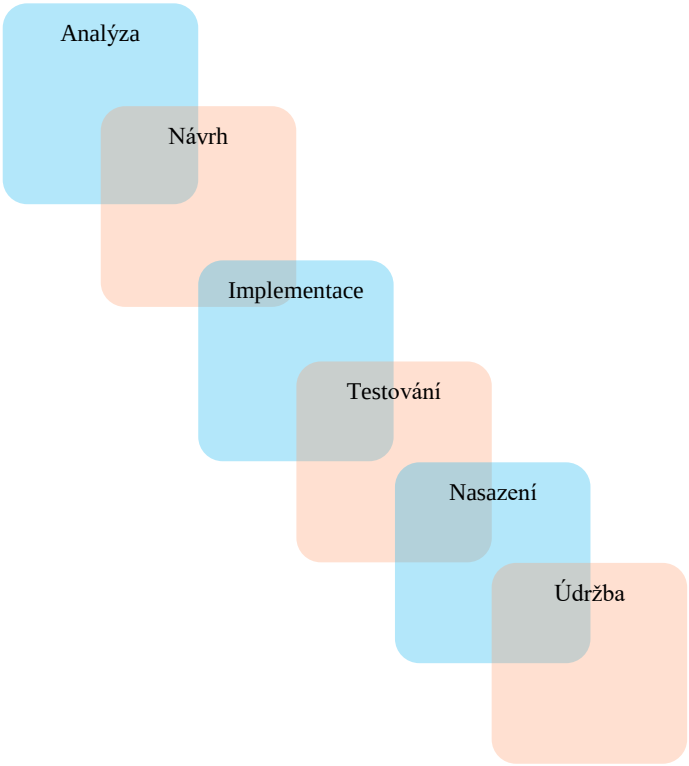
2.1.3 Sashimi model

Model známý jako Sashimi (nazvaný podle japonského pokrmu, resp. jeho způsobu podávání) je typickým příkladem modifikace Waterfallu. V literatuře ho můžeme nalézt v dříve představené publikaci „The New New Product Development Game", kdy autoři referují na jeho užití ve společnosti Fuji-Xerox (Takeuchi & Nonaka, 1986). Princip Sashimi spočívá v překryvu původních fází Waterfallu, čímž sice, co se týče pevné návaznosti aktivit, odporuje dodatku definice Adenowových (viz str. 15) a zaniká jedna z Parekhových předností (viz str. 15), avšak vtěsnání iterativní prvků do řešení může na druhou stranu zlepšit zpětnou vazbu. Například chyby provedené ve fázi analýzy mohou být zjištěny během návrhu, ačkoli definice požadavků ještě neskončila.

Podle Matkovice s Tumbasem je „dalším důležitým rysem Sashimi modelu odlišné zacházení s dokumentací. Vzhledem k tomu, že dokumentaci si v klasickém modelu Waterfallu vyměňují týmy po dokončení jednotlivých fází, model Sashimi zachází s dokumentací jako s unifikovaným dokumentem“ (Matkovic & Tumbas, 2010).

Alspaugh doporučuje užití Sashimi především pro středně velké projekty (členění velikosti projektu vysvětleno na straně 23), kde bude vyhovovat improvizační způsob komunikace (Alspaugh, 2019). Obrázek č. 6 nakonec ilustruje fázový přesah tohoto modelu:

Obrázek č. 6 – Sashimi model



Zdroj: vlastní zpracování

2.2 Agile

Na úvod není Agile vhodné považovat za samostatný model, neboť v sobě zahrnuje celou řadu agilních metodik (některé z nich jsou popsány v kapitolách 1 a 3). Korektní je o něm pojednávat jako o paradigmatu zastřešujícím soubor lehkých metodik. Jejich společné hodnoty a principy definuje právě Manifest (Beck, a další, 2001). V zásadě uceleně vystihuje komplexní myšlenku Agilu (viz podkapitola 1.3). Nejelementárnější znak Agilu pak odráží první princip – spojení iterativního procesu a inkrementální složky (viz str. 11).

Amblerova charakteristika v sobě zahrnuje evidentní vazbu na principy Manifestu. Ten Agile pokládá za „iterativní a inkrementální (evoluční) přístup k vývoji softwaru, který se provádí vysoce kolaborativním způsobem samo-organizujícími týmy v rámci efektivního rámce správy s *tak akorát (just enough)* procesem vytvářejícím vysoce kvalitní řešení nákladově efektivním a včasným způsobem odpovídajícím potřebám svých zainteresovaných stran“ (Ambler, 2007). Z ní vyplývá jmenovitě těchto 6 principů, uvedených na další straně:

» spokojenost zákazníků prostřednictvím časného a průběžného poskytování softwaru (iterativní a inkrementální (evoluční) přístup k vývoji softwaru)

» spolupráce mezi lidmi z byznysu a vývojáři v průběhu celého projektu (který se provádí vysoce kolaborativním způsobem)

» týmy s vlastní organizací napomáhající zdařilým architekturám, požadavkům a designům (samo-organizujícími týmy v rámci efektivního rámce správy)

» soustředěnost na technický detail a design zlepšující agilitu (s *tak akorát [just enough]* procesem vytvářejícím vysoce kvalitní řešení)

» krátká pravidelná dodávka funkčního softwaru (nákladově efektivním a včasným způsobem)

» očekávané přizpůsobování se měnícím se požadavkům během celého procesu vývoje (odpovídajícím potřebám svých zainteresovaných stran)

I když se vymezení snaží jednotně zahrnout mnoho principiálních skutečností, vycházet z něj samostatně by nebylo úplně vhodné. Vytvořit precizní definici Agilu představuje vskutku nelehký úkol, protože stačí přidat pocitový důraz a daná formulace dokáže spíše znejistit. Ambler zesiluje smysl spolupráce, když zmiňuje *vysoce kolaborativní způsob*. Na druhou stranu Cockburn hovoří o tom „být efektivní a manévrovatelný. Užívat snadná, ale dostatečná pravidla v projektovém chování a užívat lidská a komunikačně orientovaná pravidla," (Cockburn, 2001) takže k Agilu pojí obzvlášť schopnost komunikace. Která z těchto oblastí tedy zasluhuje větší pozornost – spolupráce, nebo komunikace?

Abrahamsson se spoluautory shrnují, co činí vývojovou metodu agilní. Takovému softwarovému vývoji dávají přívlastky (Abrahamsson, a další, 2002):

- inkrementální – malé softwarové releasy s rychlými cykly,
- kooperativní – zákazníci a vývojáři konstantě spolupracují a úzce komunikují (staví tím spolupráci a komunikaci na přímou rovinu),
- přímočarý – metodu je samo o sobě snadné se naučit a modifikovat, je dobře zdokumentována a
- adaptivní – schopnost uskutečňovat změny na poslední chvíli.

Kromě předchozích definic se lze setkat s Goldmanovým pojetím, kdy agilitu považuje za „komplexní odpověď na obchodní výzvy profitovat na rychle se měnících, neustále se tříštících globálních trzích s vysoce kvalitními, vysoce výkonnými a uživatelsky konfigurovatelnými službami a zbožím". Agilu rozumí jako způsobu k dosažení úspěchu a vítězství v konkurenčním prostředí. Přitom říká, že „nejde o to zvýšit efektivitu," (Goldman, a

další, 1995) což se zdá být v přímém rozporu s posledním principem Manifestu – pravidelnými úvahami o zlepšení efektivity, jenže Goldmanův názor má také pravdu. Jen je nutné ho chápat z byznysového hlediska.

Některé plusy agilního řízení jsou po předchozím rozboru v práci snadno odvoditelné. Za nejvýraznější mohou být uvedeny:

- ✓ existence adaptivního týmu, který je připraven reagovat na průběžně se měnící požadavky klienta,
- ✓ optimalizace dokumentace a
- ✓ v případě splnění předpokladů Agilu dodání vysoce kvalitního softwaru v krátké době spokojenému zákazníkovi.

Oproti tomu má Agile své nezpochybnitelné limitace. Jakkoli je při současné oblibě prezentován jako nejlepší způsob k chápání softwarového vývoje (zejména jeho metodika Scrum), není tomu tak, protože:

- × výstup projektu má pouze binární podobu (dodání/nedodání softwaru),
- × schází-li agilní smýšlení v týmu, může se to negativně odrazit např. na nedostatečném důrazu při tvorbě dokumentace nebo návrhu a
- × především u velkých projektů je složité posoudit úsilí na začátku SDLC.

Jak bylo naznačeno, současným úskalím Agilu je, že se stává buzzwordem (módním, nadužívaným a do jisté míry vyprázdněným pojmem), pro softwarové inženýrství a projektového řízení zdánlivě univerzálním řešením, které má tendenci se zhmotnit až do podoby razantních změn ve strategiích společností. Opět se jedná o potvrzení složitosti pochopení Agilu. Zkušení praktici jsou proto opatrní a doporučují jeho užití hlavně zralou úvahou.

Jelikož je historické hledisko agility ozřejměno v první kapitole včetně rozboru Manifestu a třetí kapitola přímo rozebírá jednotlivé agilní metodiky (nejvíc pak Scrum), následuje jeho srovnání s modelem Waterfallu.

2.3 Waterfall x Agile

I když Waterfall reprezentuje model a Agile paradigma, jejich komparace se přímo nabízí. Bylo by pochopitelně chybné se snažit dopracovat k přesvědčení, že jeden přístup je obecně lepší než druhý. Tuto mýlku se snad podařilo vyvrátit v předchozích podkapitolách.

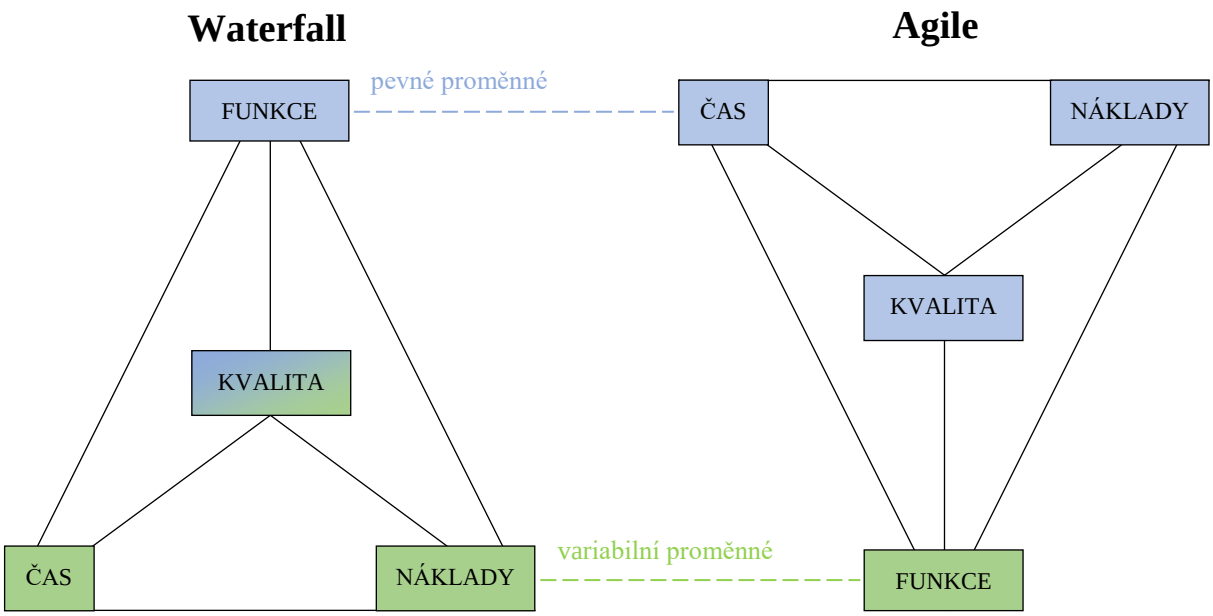
Základ pro komparaci spočívá v rozpoznání rozdílů mezi trojimperativy, čímž se zabývá navazující odstavec. Po něm jsou porovnávány charakteristiky těchto pojetí i s ohledem na dříve vyjádřené skutečnosti, vysvětlen další stupeň srovnání přes různé rozlišovací matice, a nakonec nastíněno téma Hybridu jako alternativního spojení obou pojetí.

2.3.1 Projektový trojimperativ

Pro projekty obecně platí, že jde o nalezení rovnováhy mezi třemi omezeními, tudíž se užívá termín projektový trojimperativ (triple constraint) znázorňující (formou trojúhelníku) rozpoložení času, rozsahu a nákladů (jednotlivé vrcholy). Jako čtvrté omezení figuruje kvalita (uprostřed tvaru), která má úzkou vazbu na scope (zřejmě proto se ujala o něco později).

Každý z přístupů se řídí jinou fixací vrcholů projektového trojúhelníku. Tradiční metody se zaměřují na scope projektu a na jeho harmonogram – za účelem zachování plánovaných nákladů a předpokládané doby trvání projektu, zatímco agilní metody cílí na doručování hodnoty a kvality. Za konstantní faktor ve Waterfallu platí pak scope, přičemž proměnnými se stávají cena, čas a částečně i kvalita daného softwaru. Agile zůstává konstatní v čase, nákladech i kvalitě – scope může nejen měnit, ale také očekávat jejich změny. Oba trojúhelníky a jejich omezení ukazuje obrázek č. 7:

Obrázek č. 7 – Srovnání projektových trojimperativů: železný a agilní trojúhelník



Zdroj: vlastní zpracování

Parametry projektu jsou různě pojmenovávány – na příkladu výše byl scope nahrazen slovem funkce, což lépe zapadá do kontextu softwarového vývoje. Vhodným ekvivalentem pro čas by mohl být plán, pro náklady zase zdroje, rozpočet atp.

Cline objasňuje, proč v Agilu není preferováno zobrazení původního trojúhelníku, tedy proč je vzhůru nohama, „protože to (klasické moderní pojetí á la Waterfall) odkazuje na tradiční prediktivní praktiky životního cyklu“ (Cline, 2015).

2.3.2 Charakteristické rozdíly

Špundak ve svém článku primárně řeší možnost propojení tradičního a agilního projektového řízení, nicméně analýza tzv. Hybridu (viz str. 28) musí stavět na dokonalé znalosti rozdílů těchto dvou koncepcí. Srovnání provádí přes společné charakteristiky, jak vystihuje tabulka č. 1 (Špundak, 2014):

Tabulka č. 2 – Rozdíl charakteristik mezi tradičním a agilním přístupem

CHARAKTERISTIKA	TRADIČNÍ PŘÍSTUP	AGILNÍ PŘÍSTUP
POŽADAVKY	jasné počáteční požadavky; nízká míra změn	nejasné požadavky; inovativní, kreativní
UŽIVATELÉ	nejsou účastníky	blízká a častá spolupráce
DOKUMENTACE	požadována formální dokumentace	předpokládána tichá znalost
VELIKOST PROJEKTU	vetší projekty	menší projekty
ORGANIZAČNÍ PODPORA	používá stávající procesy; větší organizace	připravenost akceptovat agilní přístup
ČLENOVÉ TÝMU	nejsou zdůrazňováni; předpoklad fluktuace; decentralizovaný tým	menší sjednocený tým
KRITIČNOST SYSTÉMU	vážné následky při selhání systému	méně kritické systémy
PROJEKTOVÝ PLÁN	lineární	komplexní, iterativní

Zdroj: Špundak, 2014

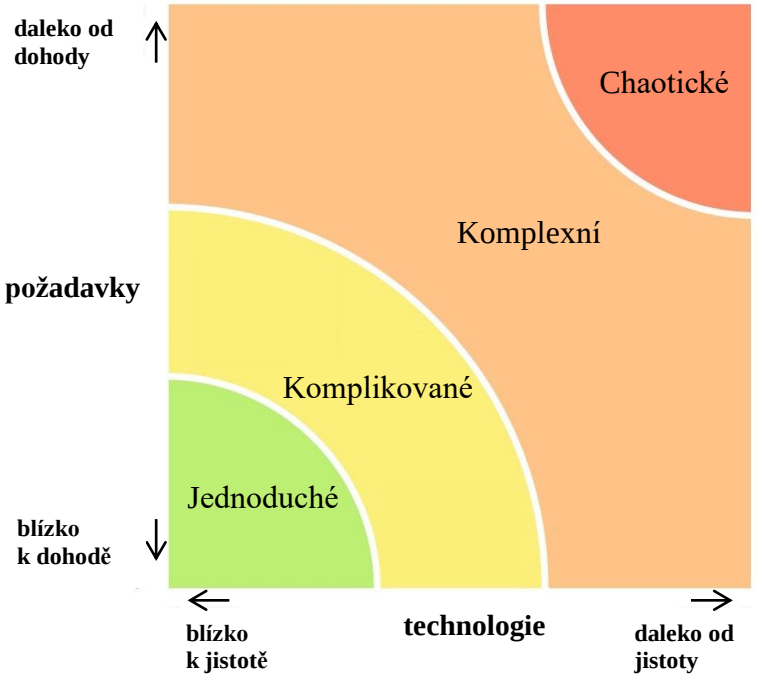
V případě tiché znalosti se myslí expertní znalost členů týmů, díky níž se redukuje potřeba vytvářet nadbytečné dokumenty. Velikost podniku je sice do jisté míry abstraktní pojem, ale na základě porovnání různých projektů dle principu trojimperativu lze projekty kategorizovat kupříkladu na menší, střední či velký (hranice se určuje intuitivně). Významnou roli hraje také rizikovost projektu (v tabulce kritičnost systému), kdy Waterfall má tendenci mnohem důsledněji pojmout řízení risků (opět propojeno s náročností na zhotovování dokumentace či zavedenými procesy ve firmě).

Rozdělení doplňuje přehled výhod a nevýhod jednotlivých metod v průběhu práce (str. 4, 15 a 22). Ty vytváří mnohdy dojem, že jsou si protikladné. Částečně ano, pokud se vztáhnou k náležitým charakteristikám, třeba aktivitě zákazníka na projektu. Například u dokumentace to neplatí – domnívat se, že ji lze v projektu vedeném agilně vynechat a spolehnout se tak výlučně na onu tichou znalost v agilním týmu, by bylo velmi idealistické.

2.3.3 Rozlišovací matice

Zajímavou paralelu pro srovnání Waterfallu s Agilem zprostředkovává Staceyho diagram (nebo též Staceyho matice). Stacey tento model vyvinul pro přiblížení komplexních situací v managementu (Stacey, 1996). Na vertikální ose se zjišťuje vzdálenost dohody (na požadavcích – co dělat) a na horizontální záruku jistoty (co se týče technologie nebo platformy – jak to udělat), viz matice níže:

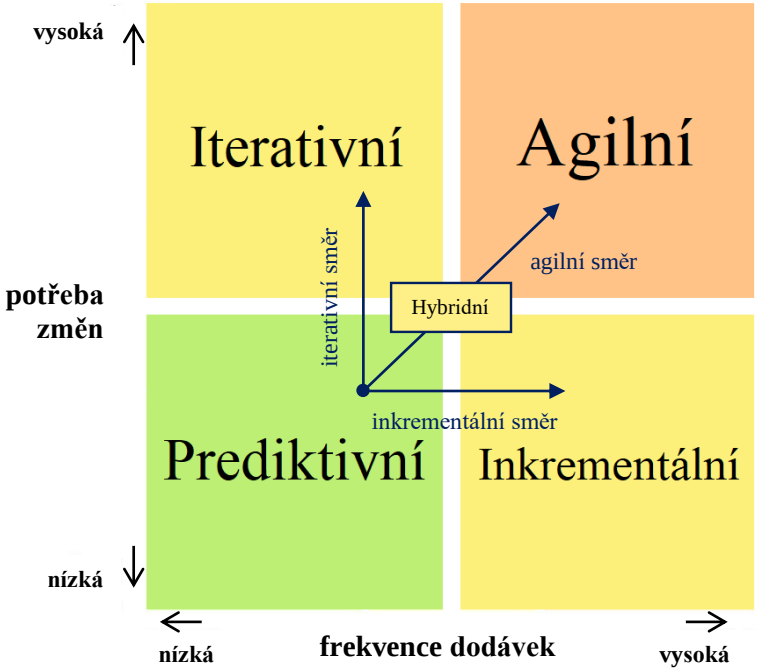
Matice č. 1 – Staceyho diagram Zdroj: vlastní zpracování



Zelená plocha zahrnuje projekty, u kterých je patrné, co dělat a jak postupovat. Pokud nastane nesoulad s požadavky, dají se užít iterace, nebo zvětšuje-li se složitost technologií, vhodným instrumentem se stává přírůstkový cyklus. Oba ukazatele posléze směřují k oblasti komplikovanosti (žlutě vybarveno). Jestliže spletitosti požadavků a technologická náročnost nabývají ještě vyšší distance (oranžová sféra), agilní metody se jeví v komplexním prostředí jako nejlepší řešení. Červená zóna chaosu pro řešení formou projektu nepřichází v úvahu.

Matice č. 2 skýtá východisko pro matici zachycenou v příručce PMBOKu (Project Management Body of Knowledge), mezinárodně uznávaného standardu řízení projektů institutu PMI (Project Management Institutute) (Lester, 2017):

Matice č. 2 – Rozlišovací matice přístupů



Zdroj: vlastní zpracování

Prediktivní kvadrant zastupuje hlavně Waterfall. Pro připomenutí: iterativní pojetí znamená opakování cyklů, kdežto inkrementální přístup dělí systémové funkcionality kouskovitě v čase. Jejich spojení tvoří dohromady základ agilních (adaptivních) metod. Porovnání slouží zejména křížově, tj. mezi prediktivními (Waterfall) a agilními metodami (Agile).

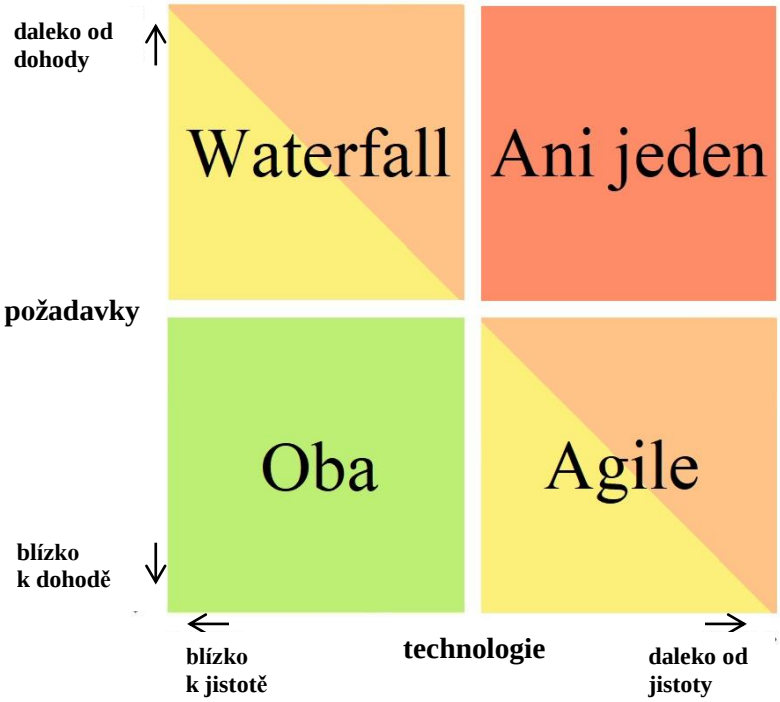
V této matici se pojmenování os od Staceyho diagramu liší – jsou blíže specifikovány, a to dle dvou charakteristik: míry změnového managementu a četnosti dodávek. Podle toho odpovídá i užitá barevná paleta (na modelu předchozí matice: zelená symbolizuje nízkou

hodnotu, zatímco oranžová vysokou). Opět souhlasí, že Waterfall není na změny koncipovaný a o frekvenci dodávek nemůže být řeč, protože se s výstupem počítá až ke konci projektu. Naproti tomu Agile obě kvantity potřebuje vysoké, a to na základě tolik protežovaného spojení iterativní a inkrementální složky.

Jednotlivá kontinua (směry) naznačují, jakým přístupem by se měl projekt řídit v případě navýšení jednotlivých hodnot. Na pomezí agilního směru se nachází hybridní metody (viz další strana).

Alternativní kombinaci obou matic prezentuje Kekstova matice (viz matice č. 3). Vznikla jako reakce na nesrovnalost v užívání rozlišovací matice přístupů s ukazateli Staceyho diagramu (vzdálenosti dohody a záruky jistoty). Kekst k tomu dodává: „Z mých zkušeností doporučuji, aby měla metodologie Waterfallu své místo, a to v levém horním kvadrantu matice. Nižší pravý kvadrant je vhodnější pro Agile. Horní pravý kvadrant (chaos) je pro obě metody stejně špatný a levý dolní kvadrant (jednoduchost) je stejně vhodný pro oba“ (Kekst, 2014). Úplně vyřadil samostatné iterativní a inkrementální přístupy, takže se čistě zabývá užitím Waterfallu a Agilu. Řadí je na pomezí komplikovanosti a komplexity.

Matice č. 3 – Kekstova matice



Zdroj: vlastní zpracování

2.4 Hybrid

Na pomezí prediktivních a adaptivních metod se formují hybridní modely. Vzhledem k tomu, že velké projekty v tradičních organizacích pod agilním vedením začaly selhávat, vznikla tendence vytvořit kombinaci tradičních a agilních způsobu a postupů v projektovém řízení (Boehm & Turner, 2004). Hybrid je přesto doposud nepříliš prozkoumaná oblast, na níž existují smíšené názory ohledně reálné použitelnosti a využití.

Podle Boehma a Turnera existuje pětice kritických faktorů na posouzení vhodnosti tradičního či agilního přístupu (Boehm & Turner, 2004):

- velikost projektového týmu
- důsledky selhání (kritičnost)
- stupeň dynamiky nebo nestálosti prostředí
- způsobilost personálu
- slučitelnost se zavedenou kulturou

Faktory jako členové týmu či kritičnost systému použil později Špundak ve své tabulce srovnání charakteristik (viz str. 24), rovněž při řešení stejné problematiky. Za další činitele lze považovat z organizační perspektivy personál a kulturu podniku. Pro volbu přístupu může být určující i sám zákazník, resp. jeho schopnost sdělovat své požadavky. Špundak dále upozorňuje, že „je důležité si uvědomit, že metodika by měla být přizpůsobena projektu, nikoli naopak“ (Špundak, 2014).

Byť Hybrid představuje neprobádanou oblast, na základě několika málo případových studií (např. Imani, a další 2017) je providitelné určit jeho klady:

- ✓ užití efektivních procesů Waterfallu,
- ✓ integraci vhodných technik Agilu,
- ✓ zlepšení přesnosti řízení a informací,
- ✓ snížení projektových nákladů a
- ✓ vyšší úspěšnost projektů.

Stejně tak má Hybrid celou řadu nevýhod, mezi nimi:

- × složitou realizaci,
- × protichůdnost principů,
- × strohou oporu v odborné literatuře a
- × větší pravděpodobnost konfliktů na všech organizačních úrovních.

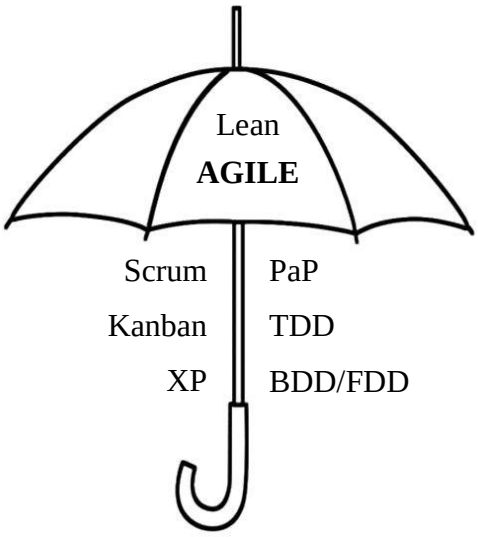
3 Agilní metodiky

První kapitola odhalila historické pozadí agilních metodik (str. 3–14) a druhá vymezila její charakteristiky v komparaci s Waterfallem (str. 15–28). Tato kapitola pokračuje klasifikací jednotlivých agilních metodik, srovnáním jejich charakteristik a postupným rozborem některých z nich – obzvlášť Scrumu.

3.1 Agilní deštník

Agilní deštník se často používá jako nástroj k zobrazení, že Agile a Scrum neznamená totéž. Jde o snahu poskytnout interaktivní a širší vhled do problematiky. Práce s danou perspektivou po celou dobu operuje, přesto je vhodné nastínit, jaké metodiky a praktiky budou na dalších stranách řešeny (viz obrázek č. 8).

Obrázek č. 8 – Agilní deštník



Zdroj: vlastní zpracování

Levá strana obsahuje tři nejčastěji užívané samostatné metodiky seřazené dle posledního „12th state of Agile reportu“ (největšího a nejdéle běžícího dotazníkového šetření k zjišťování stavu Agilu) z předešlého roku (výsledky 13. budou zveřejněny na jaře tohoto roku). Za nejběžněji užívanou metodiku mezi lehkými metodikami označili respondenti Scrum (56 %). 14 % užívá hybridních (více nakombinovaných) metodik, u kterých bohužel nelze identifikovat jejich přesný charakter. Další hybridi (pro odlišení od Hybridu [viz str. 28]: agilní hybridi) jsou již rozlišeny – Scrumban získal 8 % a spojení Scrumu/XP 6 %. Až po položce Ostatní pro nespecifikované metodiky také se 6 % se objevuje Kanban s 5% podílem. XP samo o sobě používá pouhé procento dotázaných (CollabNet, 2018).

Pravá strana obsahuje víceméně metodiky prolínající se s praktikami, proto nejsou zahrnuty ve výše zmíněném reportu. Zatím postačí rozlišit, že TDD (test driven development: programování řízené testy) je zaměřeno více na implementační aspekt systému, zatímco BDD (behaviour driven development: programování řízené požadavky na chování) na jeho funkce.

Podoba deštníku není sjednocena a často se pod ním objevuje podrobnější rozdělení na lehké a fuller (plnější) přístupy (Verma, 2017). Bylo by odvážné s ním souhlasit, jelikož všechny agilní metody jsou svou povahou lehké (normativní, flexibilní), protože jsou určeny na to, aby se přizpůsobily situaci.

3.2 Klasifikace

Přímo rámcem pro klasifikaci agilních metod se zabývá Iacovelliho a Souveyetové článek „Framework for Agile Methods Classification“ (Iacovelli & Souveyet, 2008). Z jejich tabulky (viz tabulka č. 2) vyplývá velmi podrobné srovnání vybraných agilních metodik. Níže jsou některé z nich vynechány – tabulka vychází jen z určitého výběru metodik:

Tabulka č. 3 – Srovnání vybraných agilních metodik

Proč užít metodu?	**Crystal**	**DSDM**	**XP**	**BDD/FDD**	**Scrum**
Respektování dat doručení	✓	✓	×	✓	✓
Respektování požadavků	✓	✓	✓	✓	✓
Spokojenost konečného uživatele	×	✓	×	×	×
Turbulentní prostředí	×	✓	✓	×	✓
Příznivá pro offshoring	✓	✓	×	×	✓
Zvýšení produktivity	×	×	✓	×	×
Co je součástí metodiky?	**Crystal**	**DSDM**	**XP**	**BDD/FDD**	**Scrum**
Krátké iterace	×	×	✓	✓	✓
Spolupráce	✓	✓	✓	×	×
Centralizace lidí	✓	✓	✓	×	×
Politika refaktoringu	×	✓	✓	×	×
Politika testování	×	✓	✓	×	✓
Integrace změn	×	✓	✓	×	✓
„Lehkost“ metody	×	×	✓	✓	✓

Co je součástí metodiky?	**Crystal**	**DSDM**	**XP**	**BDD/FDD**	**Scrum**
Funkční požadavky se mohou měnit	×	✓	✓	×	✓
Pracovní plán se může změnit	×	×	✓	×	×
Lidské zdroje se mohou měnit	✓	×	✓	×	×
Indikátory změn	×	×	✓	×	×
Reaktivita	milník	iterace	iterace	začátek projektu	milník
Sdílení znalostí	↑	↓	↑	↓	↓
Kdy je příznivé užít metodiku?	**Crystal**	**DSDM**	**XP**	**BDD/FDD**	**Scrum**
Velikost projektu	↑	↑	↓	↑	↑
Složitost projektu	↑	↑	↓	↑	↑
Riskantnost projektu	↑	↑	↓	↑	↑
Velikost týmu	↓	↑	↓	↑	↓
Interakce se zákazníkem	↓	↑	↑	↓	↓
Interakce s konečným uživatelem	↓	↑	↓	↓	↓
Interakce členů týmů	↑	↑	↑	↓	↓
Stupeň integrace novinek	↓	↑	↑	↓	↓
Týmová organizace	vlastní	hierarchická	vlastní	hierarchická	vlastní

Zdroj: Iacovelli & Souveyet, 2008

V tabulce jsou aplikovány tři různé pohledy:

- důvody pro užití metodiky – dané atributy mají za cíl vyhodnotit přínosy, které mohou vývojový tým a zákazník získat užitím metodiky,
- vyzdvižení vlastností podporujících agilitu – ty představují aspekty v metodice vázané na Agile a
- posouzení vhodnosti aplikace metodiky – aneb jak metoda zapadá do koloritu prostředí.

Podle kontextu znak ↓ říká, jestli je hodnota nízká/malá. Šipka nahoru ↑ obdobně označuje hodnoty vysoká/velká. Tabulka také přináší pojem offshoring, čímž je myšlena relokace procesu z jedné země do jiné, typicky operačního charakteru.

Selekce metodik plně vyhovuje potřebám práce. Cílem není zhotovit jejich kompletní srovnání, nýbrž znázornit některé rozdílné charakteristiky, které představují příhodný základ pro hlubší rozbor vybraných metodik (agilních metodik je velké množství, proto budou charakterizovány zejména ty pod agilním deštníkem ze strany 28).

3.3 Lean

Lean se původně zformoval v automobilovém průmyslu (viz str. 6). Díky tomu se užívá také doslovného překladu „štíhlá výroba“ – Lean manufacturing, odkud se pak rozšířil do celé řady dalších odvětví (např. bankovnictví, hotelnictví, zdravotnictví). Jeho koncept zdůrazňuje optimalizaci efektivity a minimalizaci odpadu (viz str. 34) v souladu s ekonomickými zásadami při vývoji softwaru. Jedná se o filozofii – širší pohled na okolní svět. Lean proto není pod agilním deštníkem řazen vedle ostatních metodik, nýbrž na jeho vrchu, protože jsou jeho principy de facto elementem Agilu jako takového.

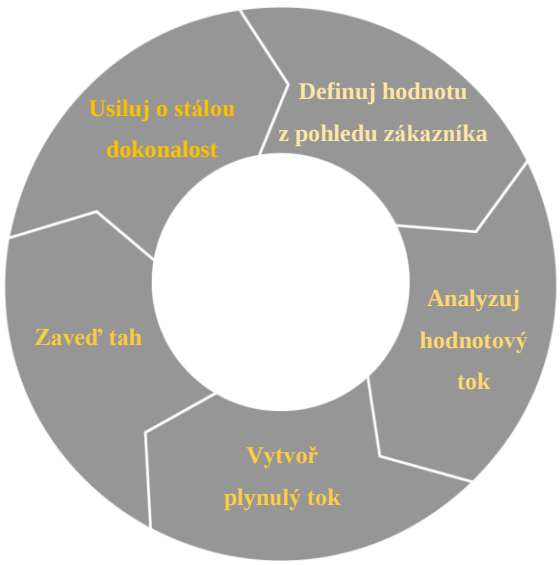
Podle Copliena s Bjørnvigem „architektura Leanu je více o zaměření a disciplíně, s podporou smysluplných argumentů, které nevyžadují žádný vysokoškolský titul ani formální vzdělání“ (Coplien & Bjørnvig, 2011).

3.3.1 Pět principů Leanu

Model pro implementaci Leanu, odvozený od Womacka a Jonese, je snadné si zapamatovat (viz obrázek č. 9), ale ne vždy jednoduché realizovat. Má za úkol napomoct podnikům vytvářet produkty zaměřené na to, co zákazníci skutečně chtějí (Womack & Jones, 1996).

Thangarajoo u jednotlivých principů postupně rozvádí jejich myšlenku. Ve zkrácené podobě jsou analyzovány na dalších stranách (Thangarajoo, 2015):

Obrázek č. 9 – Pětistupňový cyklus dle principů Leanu



Zdroj: vlastní zpracování

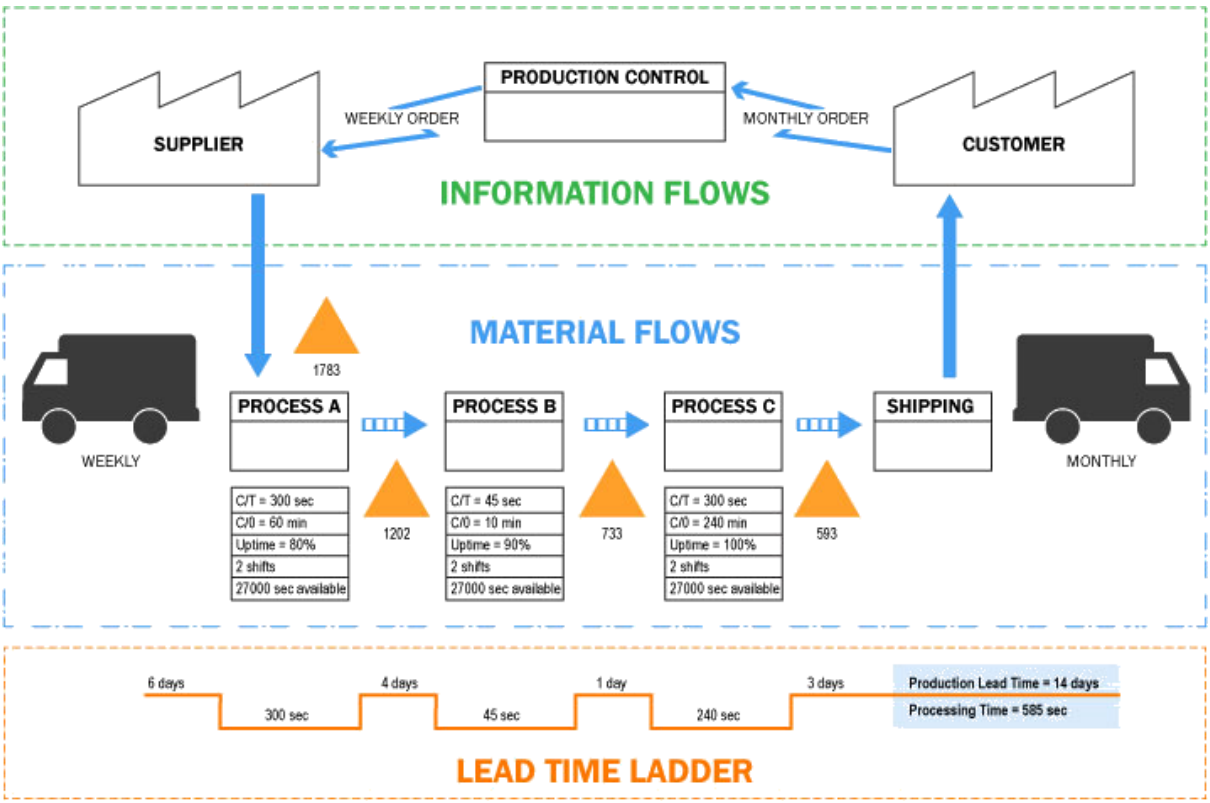
DEFINUJ HODNOTU Z POHLEDU ZÁKAZNÍKA

Princip nabádá k vyhodnocení toho, kdo je skutečným zákazníkem podniku a co hledá za hodnotu. Podniky potřebují určit hodnotu zvláště z hlediska konkrétních produktů se specifickými schopnostmi prostřednictvím rozhovoru se zákazníkem. V konečném důsledku je to totiž on, kdo rozhoduje o hodnotě produktu nebo služby. Myšlenka plně koresponduje s Agilem v oblasti spolupráce se zákazníkem (především druhý princip Manifestu, viz str. 12).

ANALYZUJ HODNOTOVÝ TOK

Mapování hodnotového toku (VSM – value-stream mapping) slouží k vizualizaci všech aktuálně potřebných aktivit přidávajících hodnotu či odpadu (viz str. 34) v rámci životního cyklu produktu nebo služby. Usnadňuje odhalování skrytých problémů v současných postupech, vytváří nové varianty pro rozhodování a zvyšuje možnost maximalizace výkonu nalezením optimálního toku v budoucnosti. Jak taková modelová mapa hodnotového toku vypadá, je uvedeno na příkladu níže:

Obrázek č. 10 – Ilustrativní mapa hodnotového toku



Zdroj: Rzepecki, 2019

VYTVOŘ PLYNULÝ TOK

Třetím principem je zavedení toku do zbývajících procesů s přidanou hodnotou po eliminaci zjevných odpadů (viz další odstavec) v hodnotovém toku. Za účelem snížení nákladů má management tendenci se soustředit na zlepšení efektivity pracovních článků (procesů) např. prostřednictvím zlepšení míry využití strojů. Kolikrát však představuje efektivnější možnost zlepšit plynulost toku, třeba zkrátit přesuny materiálu mezi procesy na co nejkratší dobu.

ZAVEĎ TAH

Další zásadu smýšlení v Leanu reprezentuje výroba založená na tahu. Mnoho procesů je totiž dáno (taženo) uživateli. V provozním prostředí je tah prostředkem k distribuci zákazníkovy poptávky po celou dobu hodnotového toku, přičemž je doručeno se pouze to, co zákazník skutečně požaduje. Přestavuje koncept, který ve výsledku závisí na přesné a aktuální prognóze poptávky za účelem stanovení parametrů produkce a úrovně zásob. Znovu je tedy významným prvkem kooperovat se zákazníkem, aby došlo k žádoucím konfiguracím na straně dodavatele.

USILUJ O STÁLOU DOKONALOST

Při respektování předchozích čtyř principů by měly být aktivity v hodnotovém toku průhlednější než dříve. Idea povzbuzuje k permanentnímu prozkoumávání nových příležitostí, aby docházelo ke zlepšování účinnosti těchto principů, neboť nikdy není stoprocentně dosaženo snížení úsilí, nákladů, času, prostoru a chyb v hodnotovém toku. Pokud podnik neodstraní všechny aktivity nepřinášející hodnoty a odpady (viz další odstavec), celý koloběh musí znovu začít (viz obrázek č. 9).

3.3.2 Toyota 3M model

Toyota v rámci Leanu zaměřuje svou pozornost zejména na odpad. Cho z Toyoty ho dle Suzakiho popisuje jako „nic jiného než minimální množství příslušenství, materiálů, dílů, prostoru a pracovní doby, které jsou pro hodnotu produktu naprosto nezbytné“ (Suzaki, 1987). Noya dodává, že „každá aktivita, která neposkytuje hodnotu, je klasifikována jako odpad“ (Noya & Chandra, 2014).

Odpad je rozlišen na tři části (3M) a pojmenován podle japonských slov „muri“ (přetížení), „mura“ (nerovnoměrnost) a „muda“ (zbytečnost, tj. nadbytečnost). Ty charakterizoval Noya zhruba takto (Noya & Chandra, 2014):

MURI

Přetížení se týká činností, které se obtížně provádí kvůli nedostatečnému vybavení, chybějícím nástrojům, neergonomickému prostředí (nevyhovujícím pracovním podmínkám) atp. Za muri je považováno také to, pociťuje-li pracovník větší míru stresu. Vědomí tohoto druhu odpadu proto napomáhá manažerům při návrhu práce či stanovení úkolů.

MURA

Jedná se o odpad z nestabilního procesu (nerovnosti ve výrobě a službách). Většinou jsou důvodem buď nestandartní pracovní postupy (případně neexistence vnitropodnikových směrnic) nebo špatně zvolená technika při analýze poptávky. Odstraňován může být manažery přes úrovňové plánování (technikou související s nastavením plynulého toku za určité období) nebo pečlivou pozorností na tempo práce.

MUDA

Muda sestává ze dvou typů činností – těch, které nemají žádnou hodnotu, ale jsou nezbytné (nevylučitelné, muda I) a aktivit, které obdobně nedisponují hodnotu, ale dá se jim vyhnout a eliminovat je (muda II). Zastupuje tudíž každou činnost, která spotřebovává prostředky bez vytváření hodnoty pro zákazníka.

Ohno označil sedm forem plýtvání (7 wastes) ve výrobním sektoru (Ohno, 1988), které jsou zahrnovány pod mudu. Mezi ně se řadí odpad z:

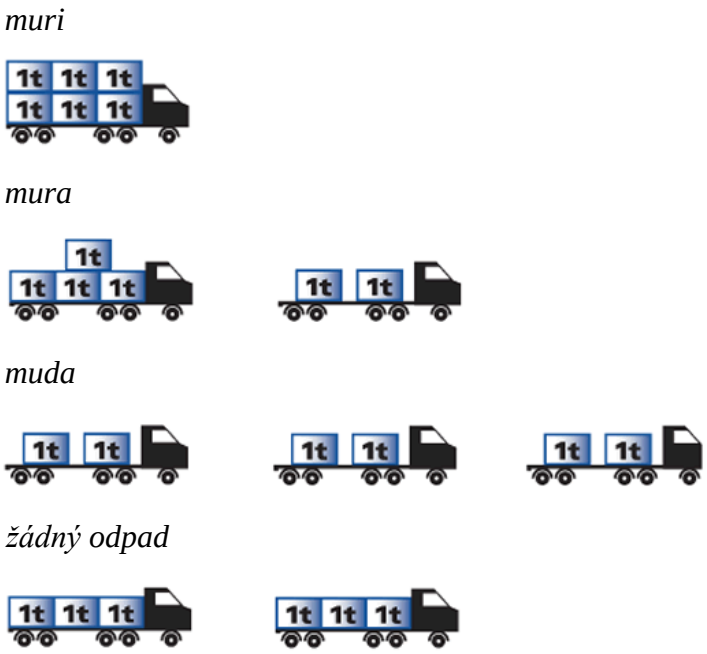
- nadvýroby – větší produkce, než odpovídá požadavkům zákazníka,
- čekání – zákazníka nebo zaměstnance, zbytečné prostoje
- přepravy – zbytečné přemisťování produktu z jednoho místa na druhé nebo mezi zaměstnanci,
- vlastního zpracování (nadbytečné zpracování) – zbytečná manuální práce, která nepřispívá k hodnotě produktu,
- skladové zásoby (inventáře) – nadbytek materiálů a informací
- pohybu – zbytečný fyzický či duševní pohyb, často spojený s hledáním a
- výroby defektů – chyby, se kterou se zákazník dostane do kontaktu.

Později byl rozpoznán osmý typ odpadu (7+1), který plýtvání spatřuje v lidech, kreativitě, motivaci a schopnostech, kdy nedochází k úplnému využití potenciálu pracovníků (Liker, 2004).

3M MODEL

Společný model 3M společnosti Toyota ilustruje obrázek níže, prezentovaný LEIem (Lean Enterprise Institutem) (LEI, 2000):

Obrázek č. 11 – Toyota 3M model



Zdroj: LEI, 2000

Pro zkoumání obchodních procesů kvůli neefektivitě se berou v úvahu všechna 3M společně proto, že jsou navzájem propojena. Aby byla vyloučena muda, většinou je nutné řešit nejprve muru nebo muri.

Jako příklad lze uvést výrobní oddělení hodnocené na základě počtu vyprodukovaných jednotek za měsíc. Projevuje se to trendem zvyšování výroby ke konci měsíce. V posledním týdnu měsíce si oddělení uvědomí, že stále nedosahuje cílového množství jednotek, a tak rapidně zvýší výrobu, než bude měsíc uzavřen. Tento strmý nárůst ve výrobě (mura) vede k tomu, že je pracovní kapitál (lidé, stroje) přetěžován (muri). Celá situace se pak schyluje ještě k nadvýrobě (muda).

3.3.3 Lean startup

Za zakladatele metodiky Lean startupu je považován Ries, který podporuje myšlenku minimalizace času a nákladů, aby mohly být společnosti efektivní. Startupu (samo o sobě širokému pojmu) rozumí „začínajícímu podniku určenému k vytvoření nového produktu nebo služby v tržních podmínkách s velkou nejistotou“ (Ries, 2011).

Jeho nápad spočívá v tvorbě hypotéz o obchodním modelu. Ty se mají testovat prostřednictvím nepřetržitých experimentů, protože tak bude možné zjistit, co zákazník skutečně chce a jestli bude podnikatelský rozvoj vůbec možný. Klíčové koncepty vycházejí

z Leanu (Agilu), zejména konstantní zpětná vazba zákazníka a iterativní přístup. Ries rovněž zdůrazňuje, aby nebyly vynaloženy prostředky na nevalidované úsilí (kvůli neznámé hodnotě pro zákazníka). Uvádí, že metoda je použitelná pro jakýkoli typ společnosti, bez ohledu na velikost či odvětví (Ries, 2011).

3.4 Scrum

Ve vztahu ke Scrumu je v první kapitole považována za průkopnickou publikace Takeuchi a Nonaky (viz str. 7). Ti tvrdí, že „tradiční sekvenční přístup či přístup „štafetového běhu“ ve vývoji produktu může být v rozporu s dosažením maximální rychlosti a flexibility. Místo toho by holistický nebo „rugbyový“ přístup – kde se tým snaží ujít vzdálenost jako jeden člověk, přihrávajíc si míč sem a tam, mohl lépe vyhovovat současným konkurenčním požadavkům“ (Takeuchi & Nonaka, 1986).

Samotný termín Scrum (zkratka z anglického „scrumage“ – mlýn) je proto ve skutečnosti metaforou pro uspořádanou formaci ragbyových hráčů, kdykoli připravených začít hrát svou hru znovu. Zdůrazňuje podstatnou roli týmové práce a další analogie mezi ragbyovým mančaftem a snahou o úspěšnost v záležitosti vývoje nového produktu.

V prvním odstavci kapitoly jsou představeny základní pilíře Scrumu, na jejichž základě přináší celkový rámec sestávající ze čtyř složek.

3.4.1 Tři pilíře empirismu – vymezení rámce Scrumu

Scrum není procesem postaveném na striktních definicích. Spíše se jedná o soubor pravidel a hodnot, který umožňuje týmu budovat vlastní agilní proces. Jeho rámec (viz další strana) je postaven na tzv. třech pilířích empirismu: transparentnosti, kontrole a přizpůsobení. Schwaber se Sutherlandem je líčí přibližně následovně (Schwaber & Sutherland, 2017):

TRANSPARENTNOST

Významné aspekty procesu musí být viditelné pro osoby odpovědné za výstup. Transparentnost vyžaduje, aby byla tato hlediska definována srozumitelně přes společný standart, např. pojem „hotovo“ by měli chápat všichni účastníci stejně.

U stakeholderů (zainteresovaných stran), např. zákazníka, generálního ředitele, individuálních přispěvovatelů aj., je transparentnost v jednáních očekávána. Jinak si nebudou moct navzájem důvěřovat a sdělovat si dobré i špatné zprávy v rámci projektu.

KONTROLA

Nemyslí se celopodniková inspekce či audit – kontrola se tyká výhradně Scrum týmu. Účastníci Scrumu často kontrolují artefakty Scrumu (viz str. 39). Postupují směrem k cíli sprintu (viz str. 41), aby odhalili nežádoucí odchylky. Kontrola by neměla nikdy překážet práci jako takové. Má přínos tehdy, pokud je prováděna kvalifikovanými kontrolory na pracovišti.

Jako příklad lze zmínit požadavky zákazníka během kontroly. Tým si nestěžuje, ale naopak se přizpůsobuje tak, že kooperuje se zákazníkem, aby otestoval nové hypotézy a objasnil požadavky.

PŘIZPŮSOBENÍ

Adaptace hovoří o neustálém zlepšování, o schopnosti se přizpůsobit na základě výsledků kontroly. Pokud kontrolor zjistí, že se jeden nebo více hledisek procesu odchyluje mimo meze a že výsledný produkt nesplňuje požadavky, proces nebo zpracovaný materiál musí být změněn. Provedení úpravy dokáže minimalizovat danou odchylku.

Důvodů, proč přizpůsobení tvoří pilíř Scrumu, je několik: zvýšení návratnosti investic na základě rozpoznání hodnoty, rychlejší uvedení na trh, zlepšení spokojenosti zákazníků a zaměstnanců atd.

RÁMEC SCRUMU

Předchozí členění pilířů navazuje na skladbu rámce Scrumu. Ten rozeznává:

3 ROLE (SCRUM TÝM)	3 ARTEFAKTY
product owner (vlastník produktu)	product backlog
Scrum master	sprint backlog
tým vývojářů	doručený přírůstek produktu
5 UDÁLOSTÍ	**5 HODNOT**
sprint jako takový	otevřenost
plánování sprintů	zaměření
opakované stand-up mítinky	angažovanost
zhodnocení sprintu	odvaha
retrospektivní pohled na sprint	respekt

Všechny tyto čtyři složky budou rozvedeny do samostatných odstavců. Pro hlubší pochopení Scrumu zasluhují jejich součásti neodmyslitelnou pozornost.

3.4.2 Pět hodnot Scrumu

ScrumAlliance (asociace certifikující obdorníky ve Scrumu) rozlišuje pět hodnot Scrumu jakožto základu jeho procesů a interakcí. Všechny byly odvozeny z principů Manifestu a jedná se o (ScrumAlliance, 2019):

OTEVŘENOST

První hodnota silně dbá na „odhalenost nebo zřejmost“. Dá se pojmout jako přístup k dostupnosti beze snah o utajení. Otevřenost v zásadě úzce souvisí s empirickým pilířem transparentnosti.

ZAMĚŘENÍ

Zaměření je o soustředěnosti a pozornosti. Pokud schází koncentrace, rozptýlení lidé nemají šanci pochopit smysluplnou úroveň hloubky řešeného. Scrum vyžaduje, aby každý člen týmu udržoval patřičnou pozornost, protože jedině tak může pracovat efektivně.

ANGAŽOVANOST

Doslovné „dejme hlavy dohromady“ zahrnuje sdílení upřímného záměru jednat a pak přijmout odpovědnost za toto zamýšlené jednání. Ve Scrumu se lidé osobně zavazují k dosažení společných cílů Scrum týmů.

ODVAHA

Hodnota říká, že má člověk jednat v souladu s individuálním přesvědčením – především, když je to složité. Odvahou se myslí vyjádřit pochybnosti, mluvit o překážkách, nebát se zeptat o pomoc, pomáhat atd.

RESPEKT

Respekt (ve smyslu pozitivní úcty) poukazuje na to, jak hodnotíme ostatní. Žádá, aby byly upřímně oceňovány jedinečné schopnosti členů týmu, kteří ho obohacují. Bez respektu pak hrozí ve Scrum týmu zánik pozitivní komunikace.

3.4.3 Tři artefakty Scrumu

Wagenaar artefakt definuje jako „hmatatelný výstup, který vzniká při vývoji softwaru... funguje jako prostředek komunikace“ (Wagenaar, a další, 2015). Scrum artefakt poskytuje klíčové informace, které musí Scrum tým a stakeholdeři znát pro porozumění produktu ve vývoji. Dosahuje také na plánované aktivity a činnosti prováděné v rámci projektu.

Mezi tři přední artefakty rámce Scrumu zpravidla patří product backlog, sprint backlog a doručený přírůstek produktu (Shipping Product Increment), všechny rozebrány níže:

PRODUCT BACKLOG

Todd product backlog (pojem backlog, tj. uspořádaná hodnota aktivních položek, se běžně nepřekládá) vysvětluje jako: „seznam pracovních položek používaný softwarovým týmem ke koordinaci práce, kterou je třeba učinit.“ Vývojáři se jím řídí – vytváří na jeho základě nové funkce, upravují ty současné a odstraňují chyby. Todd podotýká, že „mnoho týmu považuje product backlog za artefakt projektového řízení nebo primární nekódový artefakt“ (Todd, a další, 2019).

Mezi položky product backlogu patří:

- funkce:
  - user stories – uživatelské příběhy zachycující popis softwarové funkce z perspektivy uživatele
  - use cases – případ užití definující interakci mezi tzv. aktorem (působím z vnějšku, např. člověkem) a systémem
  - volná forma textu
- defekty – chyby, bugy,
- technické práce a
- získávání znalostí.

Cokoli je splněno, přestává být jeho součástí. Obsahuje tedy vše, co ještě není vyřešeno. Zobrazen je v tabulce č. 3 na další straně pro porovnání se sprint backlogem.

Řazení položek product backlogu (product backlog items – PBI) se většinou odvíjí od byznysové hodnoty (viz prioritizovaný atribut na nové straně). Jeho struktura se může řídit klíčovými atributy skrytými pod akronymem DEEP, které popisuje Pichler (Pichler, 2010):

- **D**etailed appropriately (s adekvátním detailem) – dřívější PBI by měly být více popsány, zatímco později zařazené PBI nepotřebují tolik podrobností
- **E**stimated (odhadnutelný) – přesnější odhady budou u časově bližších PBI než u těch pozdějších
- **E**mergent – (vznikající) – lépe vypovídající by bylo „dynamický“ nebo „proměnlivý“, kdy je základní předpoklad aktualizace product backlogu (přidávání, odstraňování, reprioritizování atd.)
- **P**rioritized (prioritizovaný) – více hodnotné PBI by měly být výše a méně hodnotné níže

SPRINT BACKLOG

Sprint je definován samostatně na str. 42. Sprint backlog dobře vystihuje Roudiasová, kdy o něm konstatuje, že „je souborem PBI určených pro sprint plus plán pro zajištění přírůstku podniku. Jedná se o projekci vývojového týmu, jaké funkcionality budou obsaženy v následujícím přírůstku a jaké práce jsou zapotřebí k vytvoření této funkcionality“ (Roudias, 2015). Je vytvářen na začátku každého sprintu, kdy vývojový tým přesouvá výstupy product backlogu do sprint backlogu a rozřazuje na úkoly (tasky). Každý člen má takto možnost přidávat, odstraňovat nebo měnit jeho podobu, obsah sprint backlogu by měl však zůstat po celou dobu sprintu neměnný.

Cíle sprintu popisují očekávanou dodávku na konci sprintu. Předpokládá se, že se budou řídit metodou pro stanovení cílů SMART (tj. že budou konkrétní [**S**pecific]), měřitelné [**M**easurable], přiřaditelné [**A**ssignable], realistické [**R**ealistic] a časově ohraničené [**T**ime-bound]). Příklad struktury sprint backlogu ukazuje tabulka č. 4:

Tabulka č. 3 – Zjednodušený product backlog

Product backlog (ToDo list)		
priorita	**PBI**	**odhad (v bodech)**
1	login	40
2	kontaktní stránka	20
n	**...**	**...**

Zdroj: vlastní zpracování

Tabulka č. 4 – Zjednodušený sprint backlog

Sprint backlog		
priorita	**sprint/task**	**status**
vysoká	**1. sprint**	**probíhá**
vysoká	databáze	hotovo
střední	login	probíhá
střední	**2. sprint**	**nezačal**
střední	kontaktní stránka	nezačal
n	**...**	**...**

DORUČENÝ PŘÍRŮSTEK PRODUKTU

Přírůstek produktu je výsledkem transformace požadavků do podoby doručeného softwaru během sprintu. Jedná se o fungující software (přesně tak, jak se zmiňuje druhá hodnota Manifestu, viz str. 10). Nový přírůstek je tedy implementovaný, otestovaný balík nových a plně implementovaných funkcí, testovaný a zbavený známých problémů, integrovaný a nasazený do produkce.

Produktový inkrement prakticky znamená status „hotovo“ ve sprint backlogu pod souhrnem sprintu (lze si představit v tabulce č. 4 po dokončení zhotovení loginu u prvního sprintu).

3.4.4 Pět událostí Scrumu

Události se konají ve Scrumu za účelem vytvoření pravidelnosti a minimalizace nutnosti setkání mimo těchto pět konkrétních mítinků. Každá z nich má stanovenou maximální dobu trvání, což je dané především sprintem jako takovým (viz níže) – nemohou být v průběhu samovolně prodlužovány ani zkracovány.

Kromě sprintu představují události formální příležitosti kontroly a adaptace (hlavně poslední mítink, viz str. 45). Nakonec jsou uzpůsobeny také tak, aby byly transparentní, takže v nich lze snadno identifikovat všechny tři pilíře empirismu (viz str. 37).

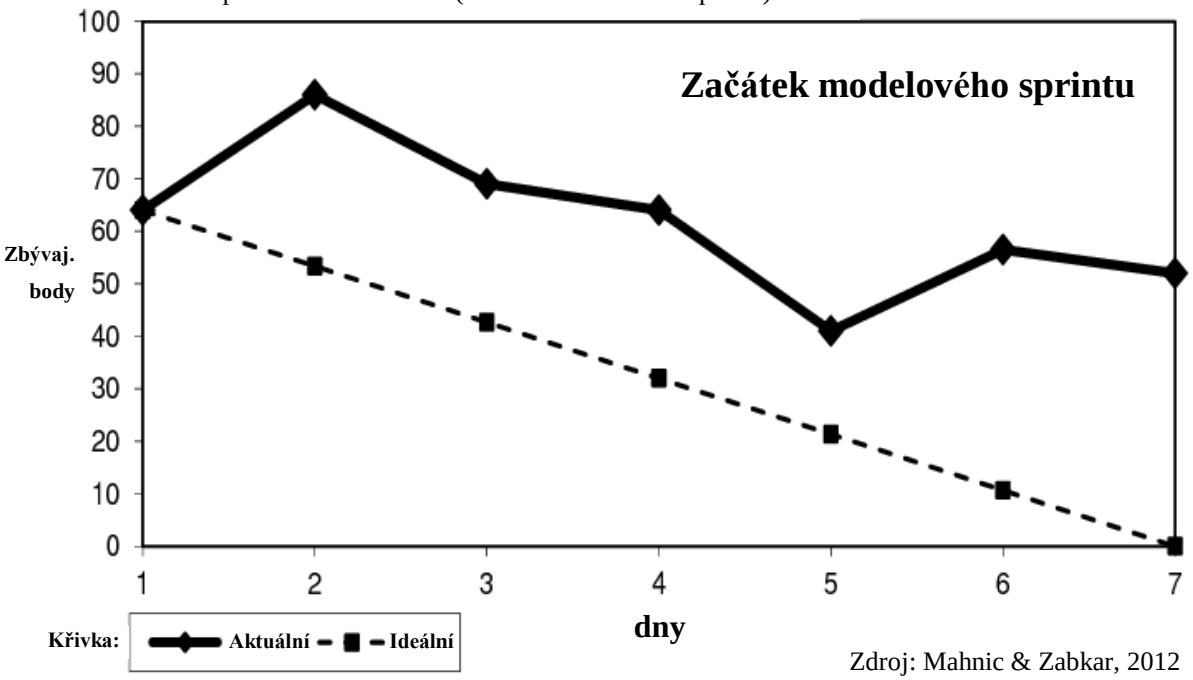
SPRINT JAKO TAKOVÝ

V první řadě je nutné se seznámit s pojmem sprint jakožto nejpodstatnější událostí, od kterého se jednak formuje myšlenka sprint backlogu (viz předchozí strana), jednak odvíjejí právě další čtyři události (viz pokračující rozbor). Schwaber ho se Sutherlandem přirovnávají k srdci Scrumu a upřesňují, že se sprintem rozumí „časově ohraničená doba o délce jeden měsíc nebo kratší, během které je vytvořen hotový, použitelný a potenciálně uvolnitelný produkt.“ Zároveň upozorňují, že během sprintu (Schwaber & Sutherland, 2017):

- nesmí být provedeny změny, pokud by měly ohrozit cíl sprintu,
- se cíle ohledně kvality nezmenšují a
- scope může být objasněn a znovu vyjednán mezi product ownerem a vývojovým týmem, až budou mít více informací.

Vhodným nástrojem pro zobrazení průběhu či výsledku sprintu je vhodné použít např. burndown chart s tzv. křivkou ideálu, viz graf na nové straně:

Graf – Sprint burndown chart (začátek modelového sprintu)



Zdroj: Mahnic & Zabkar, 2012

PLÁNOVÁNÍ SPRINTU

Dřív, než nastane sprint, se koná událost plánování sprintu (sprint planning). Je při ní respektován přístup just-in-time (viz str. 51). Tato událost umožňuje využití co nejpřesnějších informací k rozhodnutí, na čem se bude během příštího sprintu pracovat.

Roudiasová definuje časovou návaznost na sprint jako takový, když tvrdí, že trvá „osm hodin na jednoměsíční sprint." Čím je podle ní sprint kratší, tím proporcionálně ubývá i doba jeho plánování (Roudias, 2015).

Tým developerů si pokládá dvě důležité otázky:

1. co doručíme na konci sprintu? a
2. jak toho dosáhneme?

Očekává se podrobná připravenost všech tří zástupců rolí (viz str. 45). Předmětem prvního jednání bývá:

- stanovení dostupnosti týmů,
- plánování mítinků,
- výměna kontaktů a
- sestavení product backlogu (ideálně podle metody DEEP) a dostatečného množství user stories.

Plánování sprintu pak charakterizuje v návaznosti na první otázku (viz další strana):

* určení cíle sprintu – krátkého popisu doručeného přírůstku produktu prezentovaného product ownerem
* sestavení sprint backlogu
* kromě počáteční schůzky práce s defekty předchozího/předchozích sprintů – úprava cíle sprintu, sprint backlogu, požadavky atd.

Díky zamyšlení se nad druhou otázku se:

* plánuje – řeší se hrubá architektura a design, kreslí se diagramy (např. dle grafického jazyka UML) a používají se pomůcky jako bílá tabule (whiteboard), flipchart apod.
* rozčleňují jednotlivé tasky ve sprint backlogu – pro přehled se používá se k jednotlivým PBI fyzická tabulka úkolů (taskboard)

OPAKOVANÉ STAND-UP MÍTINKY

Někdy se také označují jako tzv. daily Scrum. Účastníci během takových schůzek stojí. Děje se tak z důvodu dodržení krátkého trvání mítinků (max 15 minut). Struktura může sestávat z principu „tři otázky“ a syntézy nástroje zvaného „kaizen noviny“.

TŘI OTÁZKY

Tři otázky lze podle potřeby různě modifikovat. V zásadě se vychází z intepretace:

a. Co jsem **včera** dokončil?
b. Co budu dělat **dnes**?
c. Jaké **překážky** brání mému pokroku?

Ať už bude výsledná struktura jakákoli, nejdůležitější jsou informace z odpovědí na otázky (Yip, 2016).

KAIZEN NOVINY

Japonská filozofie kaizen pochází z období války v Tichomoří a její podstata tkví v neustálém zdokonalování (se). Kaizen noviny je pracovní dokument mající formu tabulky. Identifikuje překážky, u kterých zobrazuje progres jejich řešení. Měl by být řádně aktualizován a revidován managementem. Dalším základním předpokladem je jeho čitelnost a srozumitelnost pro celý tým.

ZHODNOCENÍ SPRINTU

Co se týče událostí ke zhodnocení sprintu, kde se schází Scrum tým v kompletním složení, probíhá v nich vyrozumění stakeholderů o doručeném přírůstku produktu. Doporučená délka seance jsou čtyři hodiny. Vývojářský tým dokazuje zúčastněným, čeho

bylo během daného sprintu dosaženo. Samozřejmě se myslí souhrnná a přímá demonstrace funkcí systému a user stories (většinou na klonech z reálných údajů) ve vztahu k definovanému cíli sprintu.

Důležitou složkou události je zpětná vazba od přítomných posluchačů. Ti mohou navrhnout (product owner dokončené funkce rovnou akceptuje), jestli má být přírůstek dále vyvíjen.

Ze strany vývojářů je důležité zapojit schopnosti komunikační a psychologické schopnosti, aby svůj přírůstek co nejlépe předvedli a report ze sprintu tak byl úspěšný.

RETROSPEKTIVNÍ POHLED NA SPRINT

Mítink retrospektivního pohledu na sprint se koná po zhodnocení sprintu jako poslední mítink na konci sprintu. Z organizačního hlediska „se typická délka retrospektivního mítinku sprintu pohybuje kolem hodiny a půl... ale záleží na délce sprintu, komplexitě projektu, velikosti projektu a celkové zkušenosti týmu se Scrumem“ (Martinelli & Milosevic, 2016).

Během schůzky tým analyzuje:

- Co se ve sprintu podařilo?
- Co může být zlepšeno? a
- Co se bude chtít zlepšit v příštím sprintu?

Tým vedou otázky k dalšímu zdokonalování a motivují k celkovému zlepšení.

3.4.5 Tři role Scrumu (Scrum tým)

Scrum tým se skládá z těchto tří nositelů rolí: scrum mastera, product ownera a týmu developerů. Sestava je pevně určená, jelikož se mezi sebou náramně doplňují. V praxi se také objevuje, že jedna osoba zastává více než jednu roli, ale taková koncepce kvůli zachování Scrumem vyhrazené jednoty není příliš vhodná.

Další osoby mohou být také zapojeny do projektu, ale musí počítat s tím, že nebudou pojímány v rámci Scrum týmu – ostatně idea Scrumu o nich nic neříká.

Zvláštní postavení má zákazník. I on musí rozumět konceptu a adaptovat se, poněvadž vztah mezi ním a organizací, potažmo způsobem, jak bude dodáván produkt, se na základě rámce Scrumu kompletně mění.

Pro celý Scrum tým jsou typické dvě vlastnosti. Jedna je sebeorganizační, kdy se sám přímo řídí, a druhá mezifunkční, kdy disponuje veškerými znalostmi a potřebnými kompetencemi (Salnikov, 2017).

PRODUCT OWNER

Pichler uvádí, že je lákavé porovnávat roli vlastníka produktu se zaměřeními projektového či produktového manažera (Pichler, 2010). Nutno informativně podotknout, že ani jedna ze dvou zmíněných rolí ve Scrumu neexistuje, takže byť to k tomu svádí, každá role identifikovaná v této metodice je příliš specifická na prosté přirovnání. Product owner se svou skupinou kompetencí platí za ukázkový příklad.

Ze všech rolí má právě vlastník produktu pravděpodobně tu nejnáročnější úlohu. Zodpovídá za financování projektu během životního cyklu a stanovuje cíle projektu s požadavky. Jeho nejdůležitějším úsilím je dosažení maximalizace výstupu týmu (resp. výstupu pro každý task) založené na výpočtu návratnosti investice (ROI – return of investment).

ROI

V konkrétní konotaci Scrumu se ROI počítá podle vzorce (Milanov & Njeguš, 2012):

ROI = Byznysová hodnota / Úsilí

Jak byznysová hodnota, tak úsilí jsou definovány v product backlogu. Stanovení jednotek (mnohdy jen spíše určení číselného uspořádání – často se užívá Fibonacciho posloupnost, kde každé číslo je součtem dvou předchozích, viz str. 72) se liší v závislosti na rozhodnutí Scrum týmu. Je třeba brát v potaz skutečnost, že se výsledná hodnota ROI odvíjí z mnoha proměnných, např. od kvality produktu, jeho funkcí, ale také hodnocení trhu, tržní strategie apod.

ODPOVĚDNOSTI PRODUCT OWNERA

Sommerová a kolektiv uvádí některé jeho přidružené aktivity (Sommer, a další, 2014), má však opravdu širší spektrum odpovědností (sestaveno níže). Vlastník produktu:

- definuje vizi projektu,
- předkládá cíle projektu,
- zajišťuje:
  - hodnotu práce, kterou vykonává vývojový tým (byznysovou hodnotu),
  - viditelnost, transparentnost a pochopitelnost product backlogu (PBI) a celkově jeho správu,
- nese odpovědnost za ROI,
- určuje termíny releasů, čímž ovlivňuje čas dodávky funkcionalit,

- jasně interpretuje PBI,
- určuje prioritu PBI, aby bylo dosaženo cílů co nejlepším způsobem, ideálně podle vzorce:

  Priorita = (Pozitivní hodnota – Negativní hodnota) + Rizika + Závislosti / Úsilí
- podílí se na vyjádření k přijetí nebo zamítnutí výsledku implementace (doručení) a
- vyhodnocuje, což je kýženou hodnotou pro zákazníka, a věnuje mu dostatečnou pozornost.

SCRUM MASTER

Jestliže product owner zastává nejvíce obtížnou roli, Scrum master pak nejméně pochopenou. Scrum masterovo primární rozhraní stojí mezi vlastníkem produktu a týmem vývojářů – není to ale ani obchodní, ani technická role. Není výslovně masterem (mistrem) někoho, nýbrž rámce Scrumu (podobně jako existují mistři konkrétního bojového umění).

Scrum master je mediátorem, mentorem, učitelem a koučem. Směruje další Scrum role k tomu, aby užívaly rámec Scrumu korektně. Byť částečně zapadá do mustru manažerské role, tj. zajišťuje, aby zbytek týmu pracoval efektivně a účinně, nestará se o jeho kázeň.

VŮDCOVSTVÍ BEZ DISCIPLINÁRNÍ FUNKCE

Jedním z úkolů Scrum mastera, které jsou často zanedbávány, je péče o blaho svých členů týmu, což ovšem vylučuje liniové řízení či disciplinární akce. Z toho důvodu není doporučováno obsazovat na post Scrum mastera liniové manažery.

Disciplinární funkce odpadá z jednoho prostého důvodu – tým se sám organizuje a řídí. Scrum master zkrátka také není projektový manažer, pro kterého je daná funkce jednou z klíčových. Jeho souhrnnou vizí je vzdělávat organizaci v otázce Scrumu.

ODPOVĚDNOSTI SCRUM MASTERA

Úlohy Scrum mastera rozlišuje Schwaber se Sutherlandem na tři úrovně dle orientace podpory: vztah k vlastníkovi produktu (**P**), týmu developerů (**D**) a organizaci jako celku (**O**). Pomáhá (Schwaber & Sutherland, 2017):

- **P** ...nacházet techniky pro efektivní správu product backlogu,
- **P** Scrum týmu porozumět potřebě jasných a stručných PBI,
- **P** vysvětlit plánování produktu v empirickém prostředí,
- **P** zajistit, aby product owner věděl, jak maximalizovat hodnotu v product backlogu,

P pochopit a praktikovat agilitu a
P usměrňovat Scrum události podle potřeby.
D ...koučovat v kontextu sebeorganizace a mezifunkčnosti (viz str. 45),
D vytvářet produkt s vysokou hodnotou,
D odstraňovat překážky za účelem pokroku vývojového týmu,
D mentorovat v organizačním prostředí, které není se Scrumem dosud spjato a
D usměrňovat Scrum události podle potřeby.
O ...vést organizaci k adaptaci na Scrum,
O plánovat implementaci Scrumu,
O zaměstnancům a stakeholderům chápat Scrum, potažmo vývoj empirických produktů
O aplikovat změny vedoucí k zvýšení produktivity Scrum týmu a
O propojit zkušenosti s dalšími Scrum mastery v podniku (viz Scrum Scrumů na str. 49).

TÝM VÝVOJÁŘŮ

Pokud Scrum tým dbá na sebeorganizaci a mezifunkčnost, tým vývojářů pak dvakrát tolik. Jedná se o tým profesionálů, kteří jsou v organizaci oprávněni řídit vlastní práci. Jsou to přímí tvůrci hodnoty (inkrementu) – překladatelé požadavků do podoby fungujícího softwaru. Jsou díky tomu zodpovědni za dodanou kvalitu. Složení vývojového týmu je navrženo tak, aby se skládalo přibližně z deseti lidí.

Roudiasová jej podrobněji charakterizuje následovně (Roudias, 2015):

- nikdo (dokonce ani Scrum master) neříká vývojářům, jak proměnit PBI do přírůstku produktu nebo hypotetických funkcionalit,
- samotní developeři nemají v rámci Scrumu žádné speciální označení než developer,
- odpovědnost padá na celý tým vývojářů, i když jsou v kolektivu individuální členové specializováni a
- tým může kvazizahrnovat podtýmy věnující se zvláštním oblastem jako je testování nebo byznys analýza – všichni v týmu za takovou spolupráci ale ručí.

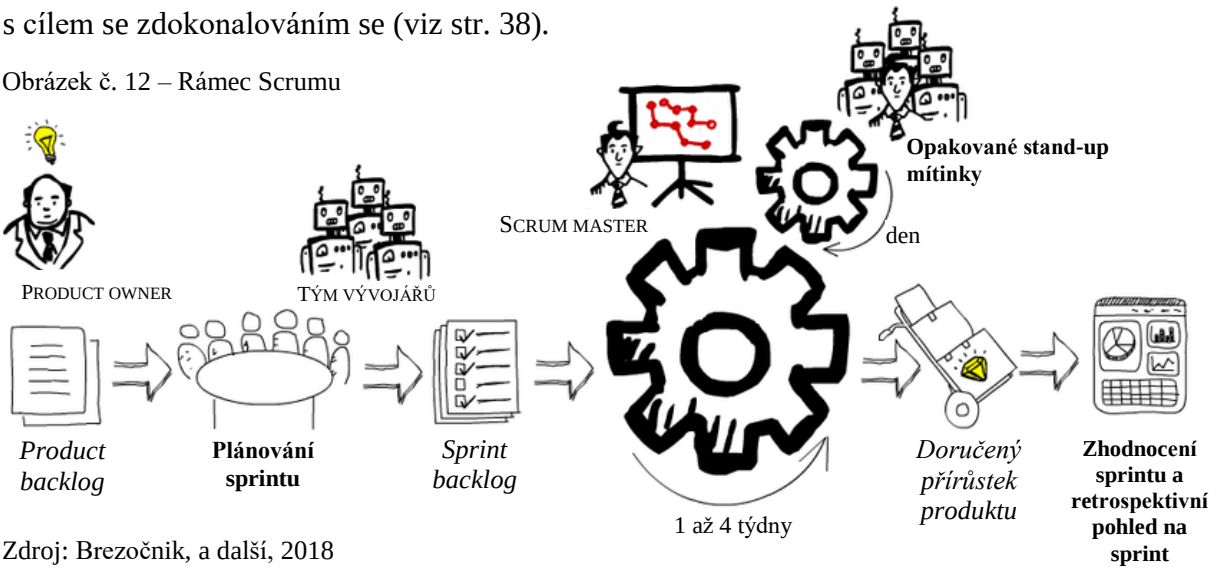
3.4.6 Shrnutí rámce Scrumu

V předchozích čtyřech odstavcích byly podrobně rozebrány základní součásti rámce Scrumu. Zbývá tedy doplnění o komplexní grafické znázornění, jak je Scrum implementován na týmové úrovni uvnitř organizace. Brezočniková interaktivně znázornila rámec Scrumu tak, jak je reflektováno na obrázku č. 12 (Brezočnik, a další, 2018).

Celý proces začíná definicí product backlogu a PBI, na což navazuje plánování sprintu a tvorba sprint backlogu. Poté přichází na řadu sprint. Každý den sprintu provází na počátku

krátký stand-up mítink. Až sprint skončí, konají se zhodnocení sprintu a retrospektiva sprintu s cílem se zdokonalováním se (viz str. 38).

Obrázek č. 12 – Rámec Scrumu



Zdroj: Brezočnik, a další, 2018

3.4.7 Scrum Scrumů

Jednu z modifikací Scrumu představuje Scrum of Scrums. Koncept skýtá řešení pro početný Scrum tým. Ten je rozdělen do menších týmů (dvou či více) respektujících požadovaný počet lidí ve Scrum týmu, které jsou hierarchicky organizovány do podoby právě Scrumu Scrumů.

Větší komplexitou tohoto přístupu mohou snadno vzniknout problémy týkající se obtížné komunikace, duplikace práce nebo nežádoucích závislostí mezi úkoly jednotlivých týmů. Paasivaarová s kolegy jako řešení definují tzv. mítink Scrumu Scrumů, kde kolektiv sestavený z reprezentantů (jeden na tým) rozjímá nad otázkami (Paasivaara, a další 2012):

a. Co udělal váš tým od doby, kdy jsme se naposledy viděli?
b. Co váš tým udělá, než se znovu setkáme?
c. Zpomaluje něco váš tým?
d. Chcete něco změnit v ohledu směřování vašeho týmu?

Frekvence události je na individuálním rozhodnutí týmu Scrumu Scrumů. Rozhoduje se o ní především podle náročnosti a velikosti projektu.

3.5 Kanban

Japonské slovo kanban znamená „karta“ a užívá se především jako plánovací systém. Základní myšlenka Kanbanu tkví v praktickém a úzce propojeném užití principů Leanu (viz str. 32). Oproti Scrumu nedefinuje žádné role ani události, má však své praktiky.

3.5.1 Tři nejdůležitější praktiky Kanbanu

Anderson a Carmichael v souhrnu o těchto praktikách sdělují, že zahrnují (Anderson & Carmichael, 2016):

- vidění práce a principů, které ovlivňují zpracování a
- zdokonalování procesu evolučním způsobem:
  - učením se,
  - udržováním a šířením užitečných změn a
  - zvrácením a tlumením neúčinných změn.

Praktik je dohromady šest. Ty nejvíc důležité pro Kanban jsou objasněny dle zmíněných autorů v následujících řádcích (Anderson & Carmichael, 2016).

VIZUALIZOVAT

K vizualizaci se používá kanban board (viz tabulka č. 5), i když ne jako jediný nástroj, aby byla vyobrazena práce a související procesy. Aby byl Kanban chápán jako systém nežli soustava toku, musí být definovány a zobrazeny:

a) závazné a dodací body a
b) limity nedokončené práce (WiP – work in progress) s cílem omezit probíhající práce v každé fázi mezi těmito body.

Tabulka č. 5 – Zjednodušený Kanban board

Backlog	ToDo list (3)	Vývoj (6)	Testování (4)	Produkce
user story 3 user story 4	user story 2 ■	■ A ■ C	■	■
	user story 1 ■■	■ B	■	■
legenda: ■ defekt ■ funkce X člen týmu				

Zdroj: vlastní zpracování

VYMEZIT WIP

Respektování a zavedení limitů na WiP transformuje systém na tahový. Nové položky v něm nejsou zahájeny dříve než po dokončení (ve vzácnějších případech přerušení) původní práce.

Optimalizace množství probíhající práce je pro úspěch zásadní, neboť vede k lepšímu času dodávek, jejich vyššímu počtu a ke zlepšení kvality. Příliš velké množství nedokončené práce generuje nežádoucí odpad (viz str. 34).

ŘÍDIT TOK

V Kanbanu by měl tok práce maximalizovat dodání hodnoty, minimalizovat dobu realizace a být co nejvíce plynulý. Občas se jedná o protichůdné představy. Vzhledem k tomu, že jsou dodávky mnohdy komplexní, vyžaduje se klást důraz na pečlivost dodržování pilířů empirismu (viz str. 37).

K pochopení zlepšení toku hodnoty je zapotřebí znát náklady na zpoždění pracovních položek. Obecně platí, že náklady na zpoždění jsou funkcí času a rychlosti změny hodnoty. Kanban rozeznává čtyři typy, jak se hodnota položek mění se zpožděním, tj. položky:

- spěšné – jsou vysoce urgentní,
- s fixním datem – mají vysoký dopad, pokud se prošvihne,
- standartní – jejich charakter je víceméně lineární, ale později ztrácí hodnotu a
- nedefinované – ač s nízkou naléhavostí, hrozí do budoucna strmým nárustem.

3.5.2 Just-in-time (kanbany)

Přístup JIT by mohl spadat rovnou pod Lean, jenže kombinace s Kanbanem je pro něj obzvlášť příhodná (viz str. 32). Společnost, která se snaží o JIT, zvyšuje efektivitu a snižuje odpadní procesy tím, že přijímá materiál tak, jak je třeba ve výrobním procesu. Opět je zde zřejmá náročnost na předpověď poptávky. JIT ovšem:

↑ zvyšuje produkci (menší chybovost), kvalitu,

↓ snižuje zásoby, mzdové náklady, WIP.

V symbióze Kanbanu a JIT hrají důležitou roli kanbany (karty). Ty podle Ikonena „působí jako tikety na řízení toku mezi pracovními stanicemi nebo procesy“ (Ikonen, a další, 2011). Slouží jako vizuální pomůcky, které vyvolávají akci. Je to jejich samotná povaha, která z nich vytváří vhodný prvek k JIT a naopak.

3.5.3 Scrumban

Asi není žádným překvapením, že Scrumban reprezentuje hybridní spojení Scrumu a Kanbanu. Je typickým představitelem agilního hybrida.

Alqudah s Razaliovou v kontextu Scrumbanu zmiňují, že „může zdokonalit Scrum vynecháním nevhodného (nefunkčního) procesu a přijmout vhodnější nástroj z Kanbanu... Scrumban podporuje kreativitu agilního týmu se záměrem nalezení nových způsobů, jak uspokojit společné potřeby“ (Alqudah & Razali, 2018).

3.6 Extrémní programování (XP)

Beck s Andresovou o XP tvrdí, že je (Beck & Andres, 2004):

- filozofií softwarového vývoje založenou na pěti hodnotách (Juric, 2000):
  - komunikace v oblastech jako:
    - unit-testing (zákazník-programátor, programátor-programátor)
    - párové programování (programátor-programátor, viz další odstavec)
    - tvorba úkolů (programátor-projektový manažer, programátor-zákazník)
  - zpětná vazba v různých intervalech:
    - programátoři poskytují okamžitou zpětnou vazbu o stavu systému
    - zákazníci mají rychlý feedback ke kvalitě svých příběhů
    - „osoba, která sleduje progres“ dává informaci, jestli projekt probíhá podle předpokládaného časového rámce
  - jednoduchost – XP vyžaduje výběr nejjednoduššího úkolu, na kterém se dá vůbec pracovat
  - odvaha – XP podporuje provádění drastických a neočekávaných akcí/opatření jako zrušení kódu nebo zbrzďování probíhajících testů
  - respekt – analogicky podobné jako na str. 39
- užitečnou sestavou praktických postupů a
- soubor doplňujících principů a intelektuálních technik pro přetlumočení hodnot do praktické podoby.

Juricová tyto principy, vycházející z hodnot a utvářející praktiky XP, dále upřesňuje (Juric, 2000):

» rapidní zpětná vazba – jakékoli učení o tom, jak být nejlepší v návrhu, implementaci a testování systému, musí být reflektováno v minutách/vteřinách,

» přijmutí jednoduchosti – zacházení s každým problémem, jako by mohl být vyřešen tou nejjednodušší cestou,

» inkrementální změny – řešení jakéhokoli problému sérií menších změn,

» osvojení si změn – zachovat většinu možností při řešení nejnaléhavějších problémů a

» kvalita práce – práce by měla být zábavou: takto je proveditelné dodat dobrý software.

Celkem dvanáct praktik XP charakterizuje Stapel s kolegy na příkladu univerzitního prostředí XP laboratoří (Stapel, a další, 2008). Pro potřeby této práce stačí podotknout, o které se jedná, viz tabulka na straně 51:

Tabulka č. 6 – Praktiky XP

PRAKTIKY XP		
BYZNYSOVÉ	**TÝMOVÉ**	**PROGRAMOVACÍ**
plánovací hra	párové programování	refaktoring
vydávaní malých verzí	společné vlastnictví kódu	prostý design
zákazník na pracovišti	udržitelné tempo	průběžná integrace
metafora	standardy kódu	testování

Zdroj: vlastní zpracování

XP praktiky jsou prakticky cíleny přímo na vývojáře a testery. XP nezapomíná ani na definici aktivit, kde se jeho zaměření také projevuje v podobě (Juric, 2000):

- kódování – pro XP základ, zdrojové kódy by měly být použity ke všemu:
  - komunikaci řešení,
  - popisu algoritmů a
  - formulaci testovacích scénářů atd.
- testování – automatizovaných testů prověřujících funkční a nefunkční požadavky, manuální testy jsou psány programátory (jednotkové) a zákazníky (funkční),
- naslouchání – rozvoje pravidel podporujících komunikaci a
- navrhování – každodenní součástí programátora v XP; v dobrém designu každý logický kousek přísluší do své oblasti.

3.7 Párové programování (PaP)

Při PaP kooperují dva programátoři na jedné aktivitě u jednoho počítače. Jeden z nich zastává roli řidiče, druhý navigátora, ačkoli se v literatuře vyskytují i kritické názory k zmíněnému dělení, např. v publikaci „Understanding the Agile Manifesto“ od Apkeho (Apke, 2015).

Řidič počítač přímo ovládá, klávesnicí a myší, zatímco navigátor (pozorovatel) sleduje práci řidiče a upozorňuje na jeho taktické a strategické chyby. Mezi taktické chyby patří chyby v syntaxu, překlepy nebo zvolení nevhodné metody. Strategické chyby se objevují, když nejsou splněny určené cíle (např. při řidičově implementaci) (Williams & Kessler, 2002).

3.8 Programování řízené testy (TDD)

Pomocí TDD dochází k řízení vývoje pomocí automatizovaných testů. Přistupuje se k němu tak, že se:

a. sepisuje nový kód, pokud automatizovaný test selhal,
b. odhalují a mažou duplicity.

Kromě toho, že výše uvedená pravidla naznačují pořadí úkolů v programování, způsobují komplexní individuální a skupinové chování s technickými projevy. Je třeba:

- vyvíjet organicky, tj. s funkčním kódem se zpětnou vazbou mezi rozhodnutími,
- vytvářet vlastní testy,
- aby vývojové prostředí reagovalo svižně na menší změny,
- aby návrh sestával z mnoha celistvých, volně vázaných komponentů.

Základ programování řízeného testy představuje cyklus *červená/zelená/refaktor* (red/green/refactor), bez něhož nelze zprovoznit žádný jiný aspekt TDD. Jednotlivé statusy znamenají:

a. **červená** (kód nepracuje) – sepsání malého testu, který nefunguje nebo není zpočátku vůbec sestaven,
b. **zelená** (kód pracuje, ale ne nejlépe) – tak rychlé zajištění chodu testu, že dokáže způsobit všechny nežádoucí projevy v procesu,
c. **refaktor** (funkce jsou dobře pokryty testy, což vede k provedení dalších změn k lepšímu) (Farcic & Garcia, 2018) – eliminace každé duplikace jen za účelem rozběhnutí testu (Beck, 2002).

Ve výsledku je snahou zlepšit vlastnosti produktu, tj. navrhnout design a určit chování systému podle očekávání klienta. Zároveň jde o zdůvodnění, že dané chování bude probíhat podle očekávání tak dlouho, dokud bude produkt využíván (Belware, 2008).

3.9 Vývoj řízený požadavky na chování (BDD)

Metoda BDD (nebo také FDD) testuje chování produktu, které očekáváme, že se zobrazí uživatelům.

První myšlenka BDD říká, že by mělo být nahlíženo z hlediska byznysu a technologie na konkrétní systém stejným způsobem, zatímco druhá poukazuje na skutečnost, že by měl

jakýkoli program mít identifikovatelnou a ověřitelnou hodnotu pro společnost (Fox, 2016). Formální definicí Northa:

„BDD je druhogenerační, tahová, multistakeholderská, multiúrovňová a vysoce automatizovaná agilní metodologie založená na principu "zvenčí-dovnitř" (outside-in). Popisuje cyklus interakcí se zdárně definovanými výstupy směřující k dodávce otestovaného a fungujícího softwaru...“ (North, 2009)

Tabulka č. 7 formuluje významy jednotlivých přívlastků a k připojuje k nim principy:

Tabulka č. 7 – Rozbor přívlastků definice BDD

přívlastek	princip	interpretace
druhogenerační		Vychází především z výsledků TDD nebo mu lze rozumět ve smyslu rozšíření.
tahová	enough is enough (dost je dost)	Není žádoucí se zahlcovat tím, co není zrovna potřeba.
multistakeholderská	deliver stakeholder value (doručení užitné hodnoty pro stakeholdery)	Vztahuje se na více zainteresovaných stran.
multiúrovňová	it’s all behaviour (všechno spočívá v chování)	Pracuje na více úrovních – např. nižší či vyšší úrovni kódu.
vysoce automatizovaná		Správná automatizace vytváří velmi rychlou zpětnou vazbu.
agilní	hodnoty Agile manifestu	viz str. 9–14
na principu „zvenčí-dovnitř“		Od představy, jak bude software vypadat pro uživatele až k jeho návrhu a implementaci vnitřní logiky.

Zdroj: vlastní zpracování

V neposlední řadě se BDD chápe jako způsob, jak přemýšlet a strukturovat testování a vývoj tak, aby byla maximalizována efektivita vývojářů (Benson, 2008).

Praktická část

Praktická a finální část práce se řídí dvěma dílčími body a hlavním cíle práce – zdokonalením aplikace zkoumané agilní metodiky Scrum ve vybrané firmě s fiktivním názvem Exam.

Z hlediska jednotlivých bodů dochází nejprve k vlastnímu odvození a interpretaci instrumentu gamifikace. Výklad je následně rozšířen o komentované příklady praktického užití. Ty se stávají dalším z mnoha podkladů pro dovršení hlavního cíle práce.

Před návrhem implementace gamifikace je charakterizována společnost Exam v nezbytných základech. Na základě polostandardizovaného rozhovoru s certifikovaným Scrum masterem je přiblížen projekt Ω, analyzován obsah interview a na základě teoretických poznatků a praktických srovnání navrhnuta vhodná podoba gamifikace pro zlepšení aplikovaného Scrumu v projektu.

1 Gamifikace

Gamifikace je určeným postupem pro zefektivnění Scrumu ve společnosti Exam. Je třeba nejprve určit, jak takový koncept pojmout a následně vycházet z úspěšných příkladů aplikace případových studií zaměřených na zvolenou agilní metodiku. Lze z nich navázat na zjištěné skutečnosti provedeného rozhovoru s certifikovaným Scrum masterem.

1.1 Definování

Existuje celá řada obecných formulací gamifikace, pro potřeby práce je ovšem nutné vytvořit novou definici splňující předpoklady pro uplatnění ve Scrumu. Gamifikace se totiž stala tak široce užívaným přístupem (využívá se např. ve zdravotnictví, prodeji, politice atd.), že zkrátka vyžaduje pevné uchopení v kontextu rámce Scrumu.

Navzdory rostoucímu vědeckému zájmu o gamifikaci vázanou na tuto metodiku se její výsledná podoba teprve formuje, a proto další vysvětlení pojmu přináší nový úhel nazírání na problematiku.

1.1.1 Předpoklady v rámci Scrumu

Všeobecným pojetím gamifikace (z anglického game – hra) se rozumí vnoření herních prvků do prostředí, které jimi přirozeně nedisponuje. Jestliže se oblast zúží výhradně na

Scrum, jediná z rolí, která ji ve vztahu ke Scrumu může využít pro dosažení lepší aplikace metodiky, je logicky Scrum master. Gamifikace má tedy dopad ve svém cílení hlavně na zbývající role Scrum týmu: product ownera a tým vývojářů.

Další uvažované kritérium musí spočívat v dobrovolnosti, neboť hraní proti vůli odporuje charakteru zábavy a rozvoje. Eventualita užití gamifikace by měla být projednána ve Scrum týmu, jestli se s navrhovaným konceptem vůbec ztotožňuje.

Gamifikace musí být zároveň vázána na vykonávanou práci tak, aby ji nenarušovala, ale naopak účastníky obohacovala o pozitivní požitek, a to také za předpokladu, že někdo vyhraje a prohraje – pomyslnou hlavní cenou by měl být progres celého týmu v problémové oblasti.

1.1.2 Hra x herní (gamifikační) prvky

Pokud jsou respektovány všechny skutečnosti z předchozího odstavce, k celkovému pojetí gamifikace je třeba si ještě položit otázku – co znamená herní prvek, jak zmiňuje obecně užívaná definice?

Hra sama o sobě disponuje celou řadou chápání. Může být pojímána z filozofického, psychologického nebo vzdělávacího hlediska. Jakkoli je na ní nazíráno, řídí se pravidly, které ji ohraničují, jinak by nemohla vůbec existovat.

Vzhledem k faktu, že jsou herní prvky nositelem zábavy (edukace), se bez nich ona hra také neobejde. V jakékoli hře jde takové elementy snadno identifikovat, ať už jich v sobě obsahuje hra více či méně.

Příklad představuje klasický šach, který je založen na modelu jeden proti jednomu, volbě strategie, konfliktních situacích atd. Dále mu mohou přidat na zábavě elementy časového omezení, hodnocení nebo stanovení jednotlivých úrovní obtížnosti – koneckonců šachové turnaje tuto dimenzi názorně rozvíjí. Charakter šachu ovšem zamezuje spolupráci mezi hráči; stejně tak je problémové vložit příběhovou složku.

Přestože generální pojetí referuje o herních prvcích a nikoli přímo o hře, implementování herních prvků zákonitě vytváří hru, a proto je gamifikace současně procesem tvorby hry v cílovém prostředí.

1.1.3 Vlastní definice

Z předchozích poznatků je proveditelné sestavení nové a širší formulace gamifikace vázané na užití v rámci Scrumu. V ohledu k této práci se gamifikací rozumí (viz nová strana):

> Nástroj vycházející z herního konceptu, který užívá Scrum master ke zdokonalení metodicky odpovídající aplikace Scrumu ostatními členy Scrum týmu, a to v paralelním souladu s jejich angažovaností a pracovním nasazením.

Jako může být pro Scrum mastera nástrojem střídmá kanbanizace (např. vizualizace přes lepící papírky na tabuli, viz str. 50), stejně tak se lze shodnout na tom, že je gamifikace dalším z prostředků k zefektivnění užívaného Scrumu.

Herní koncept vychází z úzké, až kompoziční vazby hry a herního prvku, která je identifikována výše. Herním prvkům nemohou zkrátka scházet pravidla utvářející hru a naopak – bez herních prvků hra postrádá aspekt zábavy, tj. způsob pro vzdělávání.

Další část definice odkazuje na odstavec ze strany 54. Ten jasně vymezuje vztah mezi rolemi v této problematice a upřesňuje, k čemu slouží.

Paralelním souladem je myšlen vlastní zájem zbytku Scrum týmu na účasti a zdůraznění, že gamifikace nesmí negativně narušovat jeho pracovní orientaci.

1.2 Praktické příklady

V podkapitole jsou řešeny příklady dvou studií zaměřených na praktické užití gamifikace, jmenovitě hra na hrdiny (RPG – role playing game) Scrum Hero kvalitativního charakteru, kde je interpretován modifikovaný rámec Scrumu (od rolí k událostem) a Video Scrum, jakožto podpůrná případová studie pro model hodnocení gamifikace (kvantifikační studie).

Zatímco první studie obsahuje klasický návrh gamifikace, druhá přichází se způsobem, jak kvantifikovat míru gamifikace v podniku. Obě studie jsou podrobně rozvedeny a okomentovány.

1.2.1 Scrum Hero (hra na hrdiny)

Souzové RPG Scrum Hero (dosl. hrdina Scrumu) přichází s terminologicky jinak pojmenovanou strukturou rámce Scrumu, přičemž ho obohacuje herní prvky jako postavy, skóre, žebříčky, odměny, medaile, trofeje atd. (Souza, a další, 2017).

Scrum tým představuje klan, který ovlivňují zákazníci – mystické bytosti svými přáními – požadavky. Jednotlivé role jsou označeny jako:

Tabulka č. 8 – Scrum tým ve Scrum Hero

ROLE	POSTAVA	CHARAKTERISTIKA
vlastník produktu	věštec produktu	udržuje válečníky v kontaktu s přáními mystických bytostí
Scrum master	Scrum léčitel	zodpovídá za pomoc klanu
tým vývojářů	válečníci	plní úkoly a výpravy – sprinty (viz str. 60)

Zdroj: Souza, a další, 2017; vlastní zpracování

Každý hráč disponuje dovednostmi – specifickou schopností (většinou technologicky vztaženou). Poté záleží na společnosti, jaké dovednosti bude v dané hře – projektu aplikovat.

Hra v sobě kombinuje tři způsoby skórování pro každého hráče – člena klanu. Ty jsou zpřehledněny v následující tabulce:

Tabulka č. 9 – Skórování ve Scrum Hero

	SKÓROVACÍ BODY (PT)	DOVEDNOSTNÍ BODY (SP)	ZKUŠENOSTNÍ BODY (XPE)
úroveň	jednoduchá	střední	těžká
příslušná aktivita	denní úkoly	úkoly spojené s konkrétní dovedností	celá hra
časové ohraničení	do konce dne	až do dokončení úkolu spojeného s danou dovedností	do konce hry
zvláštní vlastnost	resetování na konci dne	body připadají na jednotlivé dovednosti; hodnocení metály	hodnocení trofejemi

Zdroj: Souza, a další, 2017; vlastní zpracování

Dalšími herními prvky ve Scrum Hero jsou, jak je zjevné z posledního řádku tabulky výše, odměny trojího charakteru, a to:

- medaile – vztahují se k SP, kdy každá dovednost může být odměněna bronzem, stříbrem nebo zlatem,
- trofeje – vztahují se k výsledkům hry (XPE), kdy každý úspěch může být oceněn analogicky jako medaile a
- vlastní odměny – mohou být stanoveny na konci výprav nebo her organizací.

Artefakty také spadají do konceptu Scrum Hero, neboť takto dotvářejí prostředí RPG hry. Souzová rozeznává (tabulka na další straně):

Tabulka č. 10 – Artefakty ve Scrum Hero

ARTEFAKT RÁMCE SCRUMU	SCRUM HERO ARTEFAKT	CHARAKTERISTIKA
product backlog	**seznam přání**	**seznam přání mystických bytostí**
sprint backlog	**seznam úkolů**	**seznam úkolů výpravy**
doručený příspěvek produktu	**oběť**	**vykonaná oběť k uspokojení mystických bytostí**
schválený doručený příspěvek produktu	přijetí oběti	oběť, která byla přijata mystickými bytostmi
burndown charty	burndown mapy	mapy výprav a celé hry

Zdroj: Souza, a další, 2017; vlastní zpracování

Nakonec je kvůli Scrum Hero uzpůsobeno i názvosloví událostí:

1) sprint – výprava
2) plánování sprintu – plánování výpravy
3) opakovaný stand-up mítink – denní výzva
4) zhodnocení sprintu – výzva výpravy
5) retrospektivní pohled na sprint – klanová porada ke zlepšení

KOMENTÁŘ

Autoři studie uvádí, že v ověřovací fázi realizovaly týmy po čtyřech členech (věštec produktu, Scrum léčitel a dva válečníci) čtyři výpravy ve vybrané organizaci. Jako jediný ukazatel interpretují procentuální nárust úspěšnosti včasného nasazování do produkce, který se oproti předchozím výkonům sledovaných týmů údajně zvýšil o 55 %.

Studie toho prozrazuje, co se týče výsledků, reálně velmi málo, jelikož byla zveřejněna ve chvíli, kdy ještě výzkum nebyl dokončen. Pro další výsledky odkazuje na činnost Scrum Hero manažera (softwarového prototypu, o němž se nedá také příliš dozvědět) a na budoucí hodnocení výsledků prostřednictvím dotazníků o spokojenosti zákazníků a motivaci týmu.

Jako návrh gamifikace vyhovuje – byť (zatím) nepřináší relevantní výsledky (což mu lze vzhledem k čerstvosti tak komplexní problematiky odpustit), dozajista otevírá debatu na dané téma, např. lze koncept vůbec uplatnit (autoři tvrdí, že ano)? Skutečně povede ke zlepšení efektivity Scrumu? Tímto se nabízí prostor pro další výzkum.

1.2.2 Model hodnocení gamifikace (Video Scrum)

Případová studie Video Scrum od Gasca-Hurtadové je zhodnocena tzv. modelem hodnocení gamifikace (Gasca-Hurtado, a další, 2018). Tento model umožňuje vypočítat úroveň gamifikace zvoleného prostředí.

Návrh hodnotícího rámce udávají autoři jako výsledek analýz jiných studií, ze kterých sesumírovaly předpokládané esenciální komponenty (základy, metody, postupy) pro dosažení očekávaných výsledků. Identifikují tyto dva směry:

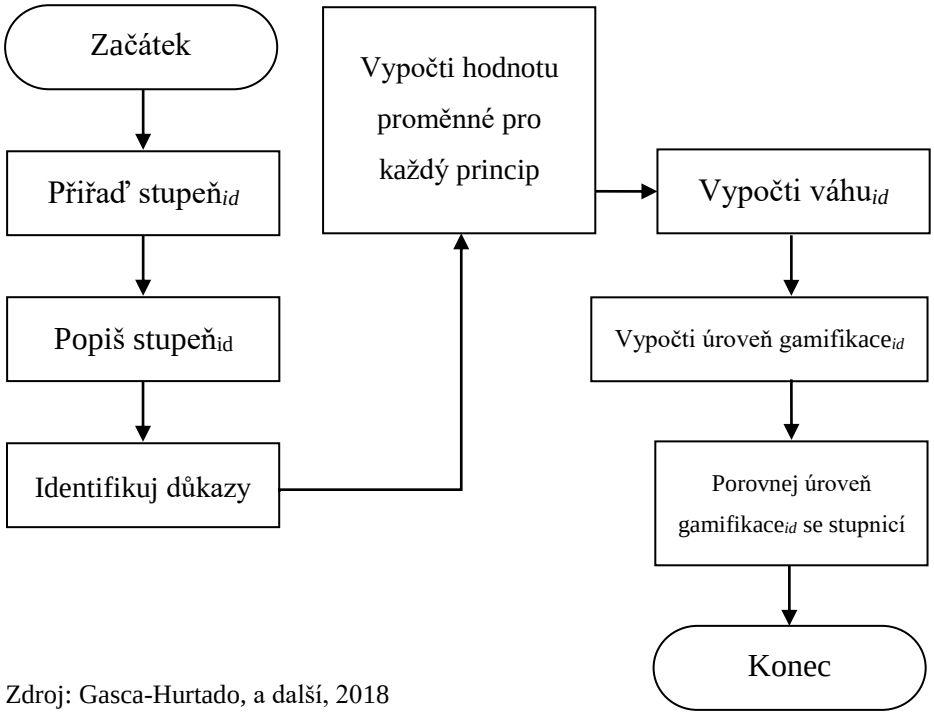
- induktivní – tvorba gamifikovaného prostředí (cílem je vytvořit gamifikované prostředí od samého počátku) a
- deduktivní – ověření splnění základů gamifikace v již gamifikovaném prostředí (identifikují se herní prvky v existujícím prostředí a pokračuje se až k odhadu) – je vhodný pro daný model.

Každý z účelů je vázán na stejnou strukturu vytvořeného rámce, tj. na následující čtyři rozlišené vrstvy:

- identifikaci principů,
- přijetí principů,
- návrh prostředí a
- hodnocení prostředí.

Gasca-Hurtadová na základě návrhu rámce sestavila postup modelu jako na uvedeném obrázku:

Obrázek č. 13 – Postup modelu hodnocení gamifikace



Zdroj: Gasca-Hurtado, a další, 2018

PROMĚNNÉ

Příslušné proměnné modelu jsou uvedeny v tabulce č. 11. Index *id* odlišuje od sebe jednotlivé principy.

Tabulka č. 11 – Proměnné modelu hodnocení gamifikace

SYMBOL	PROMĚNNÁ	CHARAKTERISTIKA
Gr_{id}	stupeň	identifikace důkazů o úrovni začlenění principů, dosažení stupně je definováno Likertovou stupnicí (1 – 5)
W_{id}	váha principu	součet posouzení důkazů vynásobený stupněm začlenění každého zjištěného principu gamifikace
LW_{id}	úroveň gamifikace	procento začlenění gamifikace do hodnoceného prostředí
Tlg	celkový počet učebních cílů	počet učebních cílů identifikovaných v herních prvcích, které jsou popsány ve vrstvě návrhu prostředí
Tru	celkový počet pravidel	počet pravidel v herních prvcích, které jsou popsány ve vrstvě návrhu prostředí

Zdroj: Gasca-Hurtado, a další, 2018

Mezi konstanty (všechny jsou obsaženy v herních prvcích, taktéž objasněných ve vrstvě návrhu prostředí) se řadí:

- *r* – role = 10
- *m* – materiály = 5
- *s* – kroky = 5

ROVNICE

Téměř každá hodnota zastupuje měřítko nezbytné pro výpočet váhy principu W_{id}. Následující přehled ukazuje rovnice pro výpočet konkrétních neznámých proměnných, a to W_{id}, LW_{id}, *Tlg* a *Tru*:

Tabulka č. 12 – Tabulka rovnic modelu hodnocení gamifikace

PROMĚNNÁ	ROVNICE
W_{id}	$W_{id} = \left(\sum Tlg + Tru + r + m + s\right) * Gr_{id}$

PROMĚNNÁ	ROVNICE
LWid	$LW_{id} = \frac{W_{id} * 100}{5\,000}$ W_{id} je zde sumou všech vah principů
Tlg	$Tlg = \frac{40}{n} * x$ n = množství učebních cílů x = množství učebních cílů pro každý princip
Tru	$Tru = \frac{30}{m} * y$ m = množství pravidel y = množství pravidel pro každý princip

Zdroj: Gasca-Hurtado, a další, 2018

VIDEO SCRUM

Případová studie Video Scrum, jak bylo naznačeno, posloužila pro ověření modelu hodnocení gamifikace, a sice deduktivním směrem, tzn. od návrhu prostředí přes pojetí principů zpět k hodnocení prostředí.

Pro sběr dat může posloužit šablona pro model hodnocení gamifikace, jako je vyobrazeno na příkladu některých prvků Video Scrumu v následující tabulce:

Tabulka č. 13 – Šablona pro model hodnocení gamifikace (vybrané herní prvky)

ID	Gr_{id}	definice	MDE	zdůvodnění	důkazy	W_{id}
1	5	Scrum tým je centrem zkušeností	mechanika	týmová spolupráce je pravidlem hry	*učební cíl č. 4* je vázán na samo-učení, samo--organizaci a samo-řízení	40
5	2	simulace Scrumu přes Video Scrum usnadňuje přijetí nového způsobu práce	dynamika	hlavní zábavné prvky ve Video Scrumu jsou: a) prostor pro kolektivní prezentaci produktu v každém sprintu b) zvýšení spokojenosti týmu, když dosáhne lepšího výsledku v retrospektivě	*krok č. 5* se týká tvorby videomateriálu sprintu a dalších událostí	10

ID	Gr_{id}	definice	MDE	zdůvodnění	důkazy	W_{id}
6	4	Scrum tým je centrem zkušeností	emoce	Video Scrum má vlastnosti zajímavé pro generaci Y: volný výběr témat a užití mobilních zařízení	*pravidlo č. 1* říká, že si Scrum tým může vybrat volné téma, jestliže moderátor představí legendu *materiály krok č. 5* se týká tvorby videomateriálu sprintu a dalších událostí	55

Zdroj: Gasca-Hurtado, a další, 2018; vlastní zpracování

Novinkou je v šabloně výše užití upřesňujícího MDE rámce Robsonové, který byl rozklíčen jako základ gamifikované zkušenosti (Robson, a další, 2015). Zkratka vychází z těchto pojmů:

- **M**echaniky – nastavení, pravidla a progrese,
- **D**ynamiky – chování hráče a
- **E**moce – stav mysli hráče.

KOMENTÁŘ

Autoři vypočetli výslednou hodnotu úrovně gamifikace Video Scrumu na 13,36 %, což podle navržené stupnice, která vymezuje gamifikaci prostředí na:

- úplnou 100 % – 76 %
- částečnou 75 % – 11 %
- žádnou 10 % – 0 %

poukazuje na prostřední variantu dle sestaveného modelu. Nutno dodat, že partikulární gamifikace je ohraničena v poměrně širokém rozmezí (interval zabírá téměř ⅔ v pravidlech měření dle studie), což je vzhledem k charakteru slovního označení víceméně pochopitelné, přesto by bylo pro lepší vypovídající hodnotu této kategorie uplatnit podrobnější členění nebo vůbec navrhnout nová pravidla měření (pojmenování, odpovídající intervaly).

Celkově je model hodnocení gamifikace vhodný pro stanovení vlivu gamifikace v návrzích (např. i pro předchozí studii Scrum Hero). Model poskytuje jak informační

hodnotu z hlediska individuálního návrhu, tak z perspektivy komparace – výsledky lze porovnat s jinými gamifikovanými prostředími, a to na poměrně detailní úrovni. Výpočet proměnných není sám o sobě náročný. Sestavení výchozí datové základny vyžaduje větší úsilí, jelikož je nutné rozvést zdůvodnění (popis herního prvku) a upřesnit důkazy (komponenty herního prvku), že se výlučně jedná o herní prvek. Díky tomu je ovšem garantován proces kontroly logicky korektního užití modelu.

2 Návrh implementace gamifikačních prvků

V závěrečné kapitole práce je referováno o provedeném polostandardizovaném rozhovoru s certifikovaným Scrum masterem (dále jen respondentem) v kontextu působení na konkrétním projektu. Na základě toho je vyhotoven návrh implementace gamifikačních prvků za účelem zefektivnění Scrumu v určené společnosti, a to v souladu s fakty uvedenými v předchozích kapitolách práce.

2.1 Rozhovor

Ke zjištění způsobu užití Scrumu ve vybraném podniku byla použita kvalitativní metoda polostandardizovaného rozhovoru, která v dialogu umožňuje tazateli vyhotovení vlastních okruhů otázek (viz příloha č. 1), ačkoli je umožněno se od struktury odklonit za účelem nalezení požadovaných informací. Povoleno je taktéž měnit pořadí otázek – záleží na vyhodnocení průběhu interview tazatelem.

Metoda byla zvolena proto, že je z principu své role nejlépe obeznámený se stavem Scrumu právě Scrum master. Ten ostatně může sám nástroj gamifikace použít k požadovanému zlepšení aplikace Scrumu, takže je tímto vypracování návrhu de facto přechýleno do prostředí této práce.

Rozhovor trval 30 minut – tj. dle plánovaného rozvržení. Přepis celého rozhovoru se nachází v příloze č. 2.

2.1.1 Anonymizace a charakteristiky

Údaje jsou anonymizovány tak, aby nedošlo k narušení pověsti firmy či samotného respondenta – přeci jen je primární snahou objevit nedostatky v interních procesech zkoumané společnosti. Proto je nutné stručně obeznámit se zaměřením společnosti, medailonkem respondenta či charakterem projektu, a až poté identifikovat nedostatky, na jejichž základě stojí návrh.

VYBRANÁ SPOLEČNOST EXAM

Exam je bezpečnostní společnost vyvíjející softwarové aplikace, osobní zařízení a řízené služby na mezinárodní úrovni. Disponuje více než třiceti výzkumnými a vývojovými středisky. Nachází se téměř v padesáti zemích světa a svoje IT služby zprostředkovává

klientům ze 150+ zemí – pro představu lze naznačit jak vlády velkých států, tak korporace, které jsou na pomyslné špičce svých oborů.

MEDAILONEK RESPONDENTA

Respondent J. W. je držitelem certifikace Certified ScrumMaster® (CSM®) od Scrum Alliance, kterou získal během působení v společnosti Exam na pozici softwarového testera (přímé testování, vytváření testovacích případů a test analýza). V korporaci pracoval přes dva roky; nyní působí především jako seniorní analytik.

PROJEKT Ω

Informace ohledně projektu Ω sdělil respondent na začátku rozhovoru. Cílem projektu bylo dodání webové aplikace pro blíže nespecifikovanou banku ze Spojených států. Tento administrativně-reklamní nástroj byl zaměřen na designování/personalizaci kreditních karet určitých klientů banky – konkrétně středně velkých podniků a profesních organizací. To znamená, že byly jejím klientům předdefinovány parametry pro vlastní design karet – bance se díky tomu zvýšil obrat používání takových karet, přičemž jejich zákazníci disponovali konkrétním podílem na zisku z užívání těchto personalizovaných karet.

Projekt byl již oficiálně ukončen, trval cca 1,5 roku (s mírným skluzem oproti původnímu plánu) a velikostně byl označen jako střední velký. Software byl úspěšně dodán. Podle Scrumu se řídil víceméně částečně, jelikož při něm došlo k postupnému přechodu z Waterfallu na Agile (Scrum), díky čemuž by se dal výsledný přístup označit jako hybridní (ve smyslu Hybridu jako na str. 28).

SCRUM TÝM (ZÁKLADNÍ INFORMACE)

Byť v projektu Ω nebyly role stanoveny striktně podle metodiky, což ve skutečnosti významně narušuje rámec Scrumu, je možné charakterizovat strukturu sledovaného týmu.

Protože se díky změně přístupů struktura týmu dynamicky vyvíjela v čase, nejprve byli v projektu identifikováni dva projektoví manažeři, ze kterých se postupem času stali vlastníci produktu; správně by měl být podle metodiky rozpoznáván pouze jeden product owner. Tým vývojářů se skládal zhruba ze 6-10 členů – byli do něj zahrnuti jak vývojáři, tak testeři, a to včetně testera/Scrum mastera (respondenta).

Mimo angažovanost v testovacích procesech dodržoval respondent svoji podpůrnou roli Scrum mastera do jisté míry v souladu s metodikou. Z kombinované pozice člena

vývojového týmu občas napomínal product owners – Scrum disciplinární funkci ovšem vylučuje (viz str. 47).

Skladba kolektivu se jeví přinejmenším chaotická. Vzhledem k metodickému přechodu se zdá být relativně pochopitelné, že došlo k transformaci rolí projektového manažera na product ownera, přesto nutno dodat, že respondent upozorňuje na zvláštní roztříštěnost odpovědnosti, kdy byla rozdělena mezi product owners a liniové manažery jednotlivých vývojářů.

EXAM A SCRUM

Ve firmě bylo rozeznáváno více Scrum týmů, ačkoli koncept Scrum of Scrums nebo jiný širší přístup nebyl uplatněn (o sdílení zkušeností nelze hovořit, maximálně ad hoc mezi vývojáři). Respondent uvádí, že z jeho pozorování reálný přístup Scrumu u ostatních týmů neodpovídal – jmenuje příklad, kdy se jednalo o detailněji rozčleněný Waterfall.

Vzhledem k silnému hierarchickému uspořádání organizace neexistovala přímá vazba mezi top managementem a daným Scrum týmem. Ačkoli snaha o zavedení rámce Scrumu byla značná, pro úplný přechod se vyskytly zábrany např. v předání odpovědnosti konkrétně na vývojáře.

2.1.2 Zevrubné skutečnosti

Idea rozhovoru byla směrována především na Scrum tým jako celek (přes oblast neformálních vztahů došlo také ke zjišťování oblastí preferovaných z herní perspektivy). Dále bylo třeba se seznámit s konanými událostmi tohoto týmu. Jako problémová se díky tomu ukázala neúčast zákazníka, která je na další straně více rozvedena.

PSYCHOLOGIE SCRUM TÝMU

Smýšlení týmu respondent pojímá jako agilní, jelikož členové spolu dlouhodobě spolupracovali (3-4 roky), a tak měli mezi sebou patřičnou důvěru. Za kladný atribut týmu vyzdvihuje zejména vědomí (ačkoli neoficiální) kolektivní odpovědnosti. Zjištěné problémy byly tak pojímány (i přes oficiální stanovení konkrétních osob, viz str. 67) v týmovém duchu (důležitý prvek Agilu, viz první hodnota na str. 10).

Tým byl díky unikátnosti projektu Ω motivován, k čemuž napomohla také možnost užití nejmodernějších technologií. Starost o stav projektu byla značná. Projevovalo se to např. formou dobrovolných přesčasů.

Vzhledem k tomu, že původní tým pocházel ze startupu zakoupeného společností Exam, příznačné pro něj byly silné (přátelské) neformální vztahy. Z nich naneštěstí nevyplynul zájem o konkrétní žánr her, přes který by se dal přizpůsobit konkrétní návrh gamifikace.

UDÁLOSTI

Konání jednotlivých událostí vesměs odpovídalo rámci Scrumu. Respondent mezi nimi vyjmenoval plánování sprintu, opakované stand-up mítinky a retrospektiva v kombinaci se zhodnocením (jediné, co by se dalo z daného hlediska vytknout – délku mítinku to protahovalo a tým pak nebyl k závěru schůzky potřebně koncentrován). K tomu se jednou týdně pořádal cca hodinový tzv. (backlog) grooming (korespondující s myšlenkou Scrumu), jehož cílem bylo upřesňování podoby PBI v product backlogu.

Sprinty měly dvoutýdenní dobu trvání, přestože jejich výsledkem nebyl inkrement v pojetí Scrumu. Ten se objevuje až jako výslednice tří samostatných částí s charakterem Waterfallu, což je opět přístup mimo rámec Scrumu.

Události byly značně narušeny tradičním charakterem společnosti (např. předimenzované členění rolí, nulová odpovědnost vývojářů atp.) a zároveň neúčastí zákazníka, který nebyl schopen akceptovat formát Scrumu. Pro tým to představovalo podstatný handicap.

KLIENT

Důležitým zjištěním se stal rigorózní postoj zákazníka na participaci, který svým přístupem ve výsledku škodil agilitě v projektu Ω. Scrum tým je připraven na změny ze strany zákazníka, nikoli však způsobem, že zákazník nashromáždí markantní množství požadavků naráz, a ještě k tomu nekomunikuje na potřebné frekvenci, což v uvedeném případě generovalo další práci na straně vývojové společnosti. Přesně proto je třeba, aby se klient pravidelně účastnil zhodnocení sprintu, protože by mělo být přirozeně v jeho zájmu, jaký výsledný produkt se mu dostaví.

Současně bylo nežádoucím faktorem zamezení dalšího fragmentu motivace, která se sice v týmu dostavovala (viz předchozí strana), nicméně výsledky práce nebylo komu prezentovat. Pozici klienta v daném případě suploval organizační útvar, aby mohlo být naplněno uznání.

Otázkou pro budoucí výzkumy je, jak přistoupit na model, v němž je klientská organizace vázána tradičním přístupem, kdežto zhotovitel operuje v agilní struktuře, případně naopak. Scrum, ač to zmiňuje nepřímo, vyžaduje od zákazníka určitý podíl aktivity na projektu – větší než v případě Waterfallu.

2.1.3 Zhodnocení rozhovoru

Z rozhovoru vyplynulo, že se v projektu Ω nejednalo metodicky o Scrum, jak je charakterizován od str. 37 této práce, nýbrž o jakýsi transformační hybrid mezi Waterfallem a Agilem.

Za zjištěné klady/zápory v projektu Ω v ohledu ke Scrumu lze z provedeného rozhovoru označit:

- ✓ agilní smýšlení týmu (vědomí si kolektivní odpovědnosti za výsledek),
- ✓ motivovanost týmu,
- ✓ individuální odbornost vývojářů a
- ✓ souhrnnou úroveň neformálních vztahů.
- × strukturu rolí týmu ignorující rámec Scrumu,
- × dopady tradiční korporátní struktury (neadekvátní rozdělení odpovědností),
- × slabá spolupráce s ostatními Scrum týmy a
- × přístup klienta (porušení frekvence dodávek).

Pokud jsou brány v úvahu zmíněné nedostatky (až na poslední z nich), vykazují interně-organizační charakter korporátního prostředí, pro jejichž řešení musí plynout úsilí z vyššího managementu organizace. Komunikačně by tomu mohl napomoct Scrum master, pokud by takovou příležitost měl – což v případě respondenta nebylo možné právě z důvodu příliš širokého vertikálního uspořádání organizační struktury společnosti.

Prostor pro Scrum mastery užít zefektivňující gamifikaci se skýtá v tvrzení respondenta, který uvedl, že ve vybrané organizaci Exam: „...byl velmi rigorózní přístup k tomu časovému plánování, a to z důvodu účtování práce ke klientovi... že se klientům musí naúčtovat přesně strávené hodiny na tom, co přesně ti vývojáři a členové týmu dělají, a potom to vede k tomu, že jednotlivé story, otázky, kroky a bugy jsou oceňované velmi rigorózně časem, který potom ne vždy odpovídá realitě, protože to jde těžko odhadnout“ (viz str. IX přílohy č. 2).

2.2 Scrum Poker

Tak, jak plyne z posouzení výroku na předchozí straně, existuje pro účely konsenzuálního odhadu se story pointy (umělá měrná jednotka) gamifikační technika s názvem Scrum Poker (plánovací poker) Jamese Grenninga (Grenning, 2002). Výběr této praktiky se dá zdůvodnit:

- ✓ přínosem přesnějších odhadů user stories, popř. jiných PBI,
- ✓ jednoduchostí,
- ✓ zaměřením na vývojový tým a
- ✓ přijatelností v podmínkách korporátního prostředí.

Její popularita je vzhledem k současné zvučnosti Scrumu na vzestupu – proto je možné využít jak fyzický materiál (balíčky karet – v prodeji často u konzultačních společností či přímo u agilních koučů), tak softwarové řešení (aplikace).

Nacházejí se v ní gamifikační prvky jako kooperace, volba nebo sdílení zkušeností. Aplikuje se konkrétně do události plánování sprintu.

2.2.1 Chronologický průběh

Jednotlivé kroky návrhu Scrum Pokeru jsou vysvětleny níže:

1. Každý člen vývojového týmu (hráč) disponuje setem karet ohodnocených dle upravené Fibonacciho posloupnosti (viz str. 46 a odstavec na další straně), eventuálně jiné. Nikomu je neukazuje.
2. Přítomný vlastník produktu (krupiér), popř. zákazník popíše uživatelský příběh a vznáší požadavek na odhad jeho délky hráči.
3. Členové týmu developerů se dodatečně informují o charakteru user story, pak stanovují vlastní odhad a po příslušné volbě pokládají před sebe kartu s hodnotou lícem dolů.
4. Karty jsou odhaleny, poprvé v zásadě s největší divergencí v odhadech. Jako konstruktivní se jeví debata vývojářů s nejnižší/nejvyšší hodnotou, jelikož svůj odhad pravděpodobně pod/nadhodnotili, zapojit se do diskuze mohou ovšem také ostatní vývojáři. Diskuze by měla mít krátkou a pevnou dobu trvání.
5. Postup předchozích kroků je opakován do té doby, dokud nedojde k žádoucí konvergenci odhadu.

Scrum master se hry účastní jako vlastník balíčku nebo správce hry v aplikaci – moderátor hry, do odhadů jako takových ale z podstaty své role nevstupuje. Jakožto navrhovatel konceptu dotváří další elementy Scrum Pokeru – zde v posledním odstavci této kapitoly.

2.2.2 Upravená Fibonacciho posloupnost

Fibonacciho posloupnost, jak bylo v práci již jednou vysvětleno, spočívá v tvorbě posloupnosti součtem dvou předchozích čísel, proto má tradiční podobu:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144...

Otázkou je, proč se používá v případě stanovení story pointů upravená verze rozvedeného číselného uspořádání. Důvodem je menší granularita vyšších čísel, resp. snaha o dosažení lepší rozpoznávací úrovně a odhadu na vysoké úrovni při přidělování hodnot k jednotlivým PBI. Příklad modifikace:

0, 0,5, 1, 2, 3, 5, 8, 13, 20, 40, 100...

Volba vyjádření jednotek se nemusí řídit striktně jen upravenou Fibonacciho posloupností. Pro vývojáře, kteří s konceptem story pointů nemají doposud zkušenost, je vhodnější aplikovat ještě jasnější měřítka, a to klidně nenumerické povahy, jako na příkladu velikosti oblečení:

XS, S, M, L, XL...

...nebo třeba ↓ nízká ● střední ↑ vysoká, kde už je ovšem granularita pro pokročilejší úroveň Scrumu příliš velká. Extrémem je pak binární varianta *0-1*.

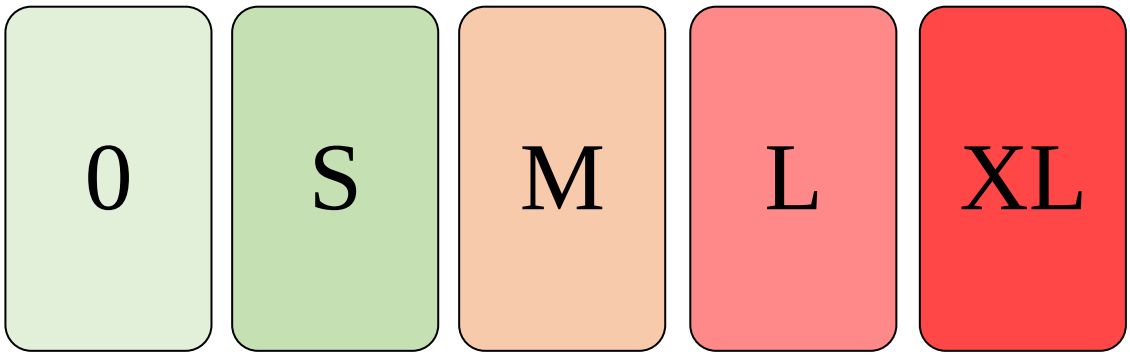
2.2.3 Vlastní rozvržení

Pro návrh gamifikačního řešení Scrum Pokeru z pozice Scrum Mastera zbývá určit podobu karet a nastavit pravidla.

PODOBA KARET

Vzhledem k tomu, že odhad story pointů ve sledovaném týmu nebyl aplikován, zvolenou variantou je jednodušší stupnice *0, S, M, L, XL* (postupem času by bylo možné přejít na upravenou Fibonacciho posloupnost s číselnými hodnotami), jako je uvedeno na příkladu obrázku na nové straně:

Obrázek č. 14 – Návrh základních karet



Zdroj: vlastní zpracování

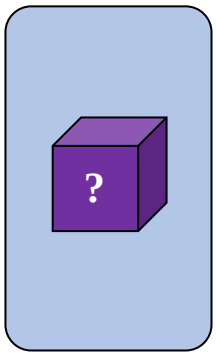
Upřesňující slovní vyjádření má pro stanovení priority user stories následující podobu:

0 – nulová (neboť takto, na rozdíl od XS, lze zahrnout také dokončené tasky)
S – nízká
M – střední
L – vysoká
XL – kritická

Speciálními kartami, které jsou v balíčcích také užívány, jsou otazník a přestávka. Otazníková karta slouží k situacím, kdy chybí znalost pro realizaci požadavku, kdežto přestávka (většinou se objevuje s piktogramem kávy) značí potřebu pauzy.

Díky tomu, že se karta s otazníkem používá jako finální volba zřídkakdy, další zamýšlenou změnou je kombinace těchto zvláštních karet do podoby karty „mystery boxu“ (tajemné krabičky, viz obrázek č. 15), která by v sobě zahrnovala všechny nestandartní situace během plánování, a tak by měla univerzální využití.

Obrázek č. 15 – Karta „mystery box“



Zdroj: vlastní zpracování

NASTAVENÍ PRAVIDEL

Kroky hry (viz str. 71) není třeba významně měnit, až tedy na nutnost časového vymezení kola doby debaty o konkrétní user story. Ta je stanovena maximálně na čtyři minuty.

Další pravidlo z hlediska času se týká posledního kroku, tj. co by mělo být impulzem k přijmutí odhadu, jelikož reálně k dokonalé konvergenci skoro nedochází a mohly by nastat nežádoucí průtahy. Hybridní charakter vedení projektu, kdy část odpovědnosti náležela na product ownerech a na liniových manažerech, vede k myšlence alespoň přechýlení odpovědnosti na vlastníky produktu ke schvalování odhadů.

2.2.4 Body konsenzu (PoCo)

Aby Scrum master napomohl tak hierarchizované organizaci se přiblížit adaptaci na Scrum, svá pozorování Scrum Pokeru může podpořit o paralelní bodovací systém, který by jednak ještě intenzivněji přiblížil vývojáře k pocitu osobní odpovědnosti, jednak podporoval snažení o hlubší diskuzi s vyšším managementem.

Vhodným způsobem se mohou stát body konsenzu (points of consensus – PoCo), jejichž získání by bylo ve hře podmíněno definitivním uznáním odhadu, a to buď hned prvního (královský PoCo ●) či (pravděpodobněji) druhého (čistý PoCo ○). V případě dalších kol by k získání PoCo dojít nemohlo. Za každou PBI by tyto body byly přidělovány **diskutujícím** vývojářům na základě pozorování Scrum masterem (přiděluje je intuitivně podle vyslovených názorů do debaty), tj. na modelu:

Tabulka č. 14 – Příklad šablony pro evidenci bodů konsenzu

PLÁNOVÁNÍ SPRINTU Č. 5			**Σ**
VÝVOJÁŘ	**PoCo**		
	●	○	
A. A.	US#2	US#4	●○
B. B.		US#1 US#4	○○
C. C.	US#2		●
D. D.		US#1	○
E. E.	US#2	US#1	○●
Schvaluje krupiér:	product owner Z. Z.		

Zdroj: vlastní zpracování

Po vyhotovení burndown chartu se vyškrtnou ty PoCo, které nebyly odhadnuty správně (příklad v tabulce č. 15 na nadcházející straně), čímž dojde k vyřazení PoCo (neplatný PoCo ●) konkrétních uživatelských příběhů. Zbytek platných PoCo je relevantních v podobě závěrečných výstupů jednotlivců pro určitý sprint. List je připraven být oficiálně prezentován Scrum masterem před zbytkem týmu formou žebříčku na události pro zhodnocení sprintu.

Dle tabulky č. 15 k nejrychlejším a nejpřesnějším odhadům přispívá vývojář A. A., zatímco rychlostí a věcností diskuze příliš ve výsledku nenapomáhá vývojář D. D. – není záměrem ho stigmatizovat, ale povzbudit ho ke snaze o lepší výsledky.

Tabulka č. 15 – Příklad šablony pro evidenci bodů konsenzu s penalizací

PLÁNOVÁNÍ SPRINTU Č. 5			**Σ**
VÝVOJÁŘ	**PoCo**		
	●	○	
A. A.	US#2	US#4	●○
B. B.		~~US#1~~ US#4	●○
C. C.	US#2		●
D. D.		~~US#1~~	●
E. E.	US#2	~~US#1~~	●●
Schvaluje krupiér:	product owner Z. Z.		

Zdroj: vlastní zpracování

Pokud by měl být příklad aplikován na prostředí sledovaného týmu z rozhovoru, nejspíš by to dotyčného vývojáře mrzelo a snažil by se z důvodu odpovědnosti na výsledku zlepšit, přičemž zbytek týmu by ho dokázal při týmové soudržnosti podpořit. Všechny tyto fakty podporují hodnoty Scrumu.

Za celý tým pak může Scrum master (s podporou vlastníka produktu) ve vyšších sférách vysvětlit princip bodování a upozornit na sesbíraný celkový počet královských PoCo a čistých PoCo – jak pro jednotlivé sprinty, tak v celkovém součtu. Je to totiž přímý důkaz dostatečné míry odpovědnosti týmu vývojářů Scrum týmu k estimaci, která se vztahuje k PBI.

Závěr

Tím, že práce obsahově navazuje na stanovené dílčí body hlavního cíle, bylo každé z řešených témat podrobně zpracováno, ať už to vyžadovalo sestavení uceleného historického vhledu do vývoje agilních metodik a hloubky Manifestu, multidimenzionální srovnání modelu Waterfallu s paradigmatem Agilu, přiblížení nejběžněji aplikovaných agilních metodik či praktik – s pochopitelným důrazem na zachycení Scrumu a celého jeho rámce, návrh vlastní formulace pro gamifikaci Scrumu, a sice s doplňujícím komentářem dvou odborných studií dotýkajících se oblastně stejné problematiky, či provedení a následné zhodnocení interview se Scrum masterem s reálnými zkušenostmi z vybrané společnosti na konkrétním projektu. Na vše ve finále navázalo vytvoření konceptu, tj. modifikované hry Scrum Pokeru s možností zaznamenávání její úspěšnosti přes tzv. body konsenzu. Splnění hlavního cíle bylo tedy zdárně dosaženo. Přínos se tak dá nalézt v mnoha oblastech, čímž záleží na preferovaném diskurzu každého.

Bylo snahou se dopracovat přímo ke stěžejním zdrojům, takže uvedená bibliografie vychází z odpovídajících, převážně anglicky publikovaných dokumentů jak v tištěné, tak elektronické podobě. Citovaný text obohatila autorská zhodnocení a upřesnění. Ta práci dodala intenzivnější teoreticko-praktický ráz.

Nejvýznamnější hodnota ovšem spočívá v rozvoji tématu gamifikace jako techniky k zefektivnění Scrumu vůbec, protože ve světě skutečně neexistuje příliš zdrojů k odbornému pochopení daného propojení. Vyhotovení této práce poskytuje záruku pro rozvoj vymezené oblasti zkoumání, jelikož v sobě zahrnuje silnou oporu v nedávných studiích a výzkumech.

Doporučení dalšího výzkumu

Téma v sobě skýtá příležitosti ke skutečně neotřelému, byť užitečnému bádání, ať už z kvantitativního či kvalitativního hlediska; tak jako tak bude nadále existovat prostor pro tvorbu nových (nebo vylepšení stávajících) návrhů implementace gamifikace. Dá se říct, že je neustále možné prověřovat zefektivnění aplikace Scrumu prostřednictvím tohoto nástroje, neboť různými kombinacemi herních prvků jde, už samo o sobě, vytvořit takřka nespočet konceptů. Výsledky mohou být při správném designu a realizaci více než příznivé. Další otázku představuje zvolený formát – v čem jsou lepší fyzické materiály oproti softwarovým řešením gamifikace (a naopak)?

Dotazů, které si lze položit ke gamifikaci Scrumu, je mnoho. Ostatně i v případě dalších metodik je proveditelné najít uplatnění, a to klidně i v případě tradičních přístupů. Zase je ale třeba vnímat, že je to nástroj, jehož užití podmiňují určité zákonitosti, aby došlo ke kýženému efektu. Nesmí se významně odchýlit od svého záměru, jinak není příliš těžké se dopracovat k plytkým, ne-li opačným účinkům.

Vzhledem k rozpětí práce se výzkum může ubírat celou řadou příbuzných směrů. Na místě je průzkum problematiky dalších samostatných modelů (V-model, Spiral). Zmíněny byly oba hybridní přístupy (Waterfall x Agile; agilní hybridi), které by při současné situaci zasluhovaly větší pozornost, než se dostává agilitě díky obecně panujícím nepravdám. Zmíněné agilní metodiky v práci nakonec nejsou jedinými; existují další jako pragmatické programování, Spotify model nebo agilní modelování.

Je tak předmětem individuálního zaměření, jakou dráhu zvolit – prostor pro výzkum se nabízí opravdu značný; primární doporučení přesto klade důraz na rozvedení gamifikace ve vztahu k přístupům, modelům, paradigmatům, metodikám atd. v (IT) projektovém řízení.

Použitá literatura

A comparative study of software development life cycle models. **Manzoor, Ahmad Rather.** 2005, International Journal of Application or Innovation in Engineering & Management (IJAIEM), str. 24. ISSN 2319-4847.

Abrahamsson, Pekka, a další. *Agile Software Development Methods: Review and Analysis.* Espoo: Otamedia Oy, 2002. str. 11–12; 17. ISBN 951–38–6010–8.

Agile Product Development Governance – On govering the emerging Scrum/Stage-gate hybrids. **Sommer, Anita Friis, Dukovska-Popovska, Iskra a Steger-Jensen, Kenn.** Heidelberg: Springer Berlin, IFIP Advances in Information and Communication Technology, str. 188. ISSN 1868-4238.

Agile software project management methodologies – prospects of the Greek IT market. **Monochristou, Vagelis, Vlachopoulou, Maro a Manthou, Vassiliki.** Constanta: University of Macedonia, 2005. The 7th Balkan Conference on Operational Research "BACOR 05", str. 3.

Alspaugh, Thomas A. Software Process Models. *Thomas A. Alspaugh.* [online] 13. únor 2019. [citace: 13. únor 2019] https://thomasalspaugh.org/pub/fnd/softwareProcess.html#Sashimi.

Ambler, Scott W. Disciplined Agile Software Development: Definition. *Agile Modeling.* [online] 1. listopad 2007. [citace: 17. leden 2019] http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm.

An Empirical Study of Scrumban Formation based on the Selection of Scrum and Kanban Practices. **Alqudah, Mashal a Razali, Rozilawati.** Bangi: 2018, International Journal on Advanced Science, Engineering and Information Technology, str. 2317. ISSN 2088-5334.

An Introduction to Agile Methods. **Cohen, David, Lindvall, Mikael a Costa, Patricia.** str. 62, 2004, Advances in computers, str. 3. ISSN 0065-2458.

Analysis of Return on Investment in Different Types of Agile Software Development Project Teams. **Milanov, Goran a Njeguš, Angelina.** 2012, Informatica Economică, str. 10. ISSN 1453-1305.

Anderson, J. David a Carmichael, Andy. *Essential Kanban Condensed.* Seattle: Lean Kanban University Press, 2016. ISBN 978-0984521425.

Apke, Larry. *Understanding the Agile Manifesto.* Berlin: Lulu.com, 2015. ISBN 978-1312863910.

Assessment Framework for Gamified Environments: A Gamification Assessment Model for Implementing the Framework. **Gasca-Hurtado, Gloria Piedad, a další.** 2018, Systems, Software and Services Process Improvement, str. 240–253. ISBN 978-3-319-97925-0.

Beck, Kent a Andres, Cynthia. *Extreme Programming Explained: Embrace Change, 2nd Edition.* Boston: Addison-Wesley, 2004. str. 2. ISBN 978-0321278654.

Beck, Kent. *Extreme programming explained: embrace change.* Boston: Addison-Wesley Longman Publishing Co, 1999. ISBN 0-201-61641-6.

—. *Test-driven Development: By Example.* Boston: Addison-Wesley Professional, 2002. str. ix-x. ISBN 978-0321146533.

Beck, Kent, a další. Manifesto for agile software development. [online] 13. listopad 2001. [citace: 17. červenec 2018] http://agilemanifesto.org/.

Belware, Scott. Code Magazine. *Behaviour-Driven Development.* [online] 1. červen 2008. [citace: 9. září 2018] https://www.codemag.com/article/0805061.

Benington, Herbert D. *Production of large computer programs. In Proceedings, ONR Symposium on Advanced Programming Methods for Digital Computers.* Boston: Massachusetts Institute of Technology, 1956. str. 15-27.

Benson, Edward. *The Art of Rails.* Indianapolis: John Wiley & Sons, 2008. str. 276. ISBN 978-0-470-18948-1.

Bitter, Rick, Mohiuddin, Taqi a Nawrocki, Matt. *LabVIEW: Advanced Programming Techniques.* Boca Raton: CRC Press, 2000. str. 152. ISBN 0-8493-2049-6.

Boehm, Barry a Turner, Richard. *Balacing Agility and Discipline: A Guide for the Perplexed.* Boston: Addison-Wesley, 2004. 978-0321186126.

Bosworth, Seymout a Kabay, M. E. *Computer Security Handbook.* Hoboken: John Wiley & Sons, 2002. str. 25.6. ISBN 0471269751.

Cline, Alan. *Agile Development in the Real World.* New York: Apress, 2015. str. 87. ISBN 1484216792.

Coad, Peter, de Luca, Jeff a Lefebvre, Eric. *Java Modeling In Color With UML: Enterprise Components and Process.* Upper Saddle River: Prentice Hall PTR, 1999. ISBN 978-0130115102.

Cockburn, Alistair. *Agile Software Development.* Boston: Addison-Wesley, 2001. ISBN 0201699699.

—. *Agile Software Development: The Cooperative Game.* Upper Saddle River: Addison-Wesley Professional, 2006. str. Supporting the Values. ISBN 0321482751.

—. *Crystal clear a human-powered methodology for small teams.* Boston: Addison-Wesley Professiona, 2004. ISBN 0201699478.

CollabNet. 12th Annual State of Agile Report. *Annual State of Agile Survey.* [online] 9. duben 2018. [citace: 3. únor 2019] https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report.

Coplien, James O. a Bjørnvig, Gertrud. *Lean Architecture: for Agile Software Development.* Padstow: John Wiley & Sons, 2011. str. 2. ISBN 0470970138.

Dana, William Harvey "Bill". William H. "Bill" Dana: X-15 Lessons Learned. *NASA.* [online] 13. červenec 2004. [citace: 11. listopad 2018] https://www.nasa.gov/centers/armstrong/history/speeches/bill_dana/X-15_lessons_learned.html.

DeGrace, Peter a Stahl, Leslie Hulet. *The Olduvai Imperative: Case and the State of Software Engineering Practice.* Ann Arbor: Yourdon Press, 1993. ISBN 978-0131611009.

Deming, William Edwards. *The New Economics for industry, government, education.* Cambridge: MIT Press, 1993. str. 135. ISBN 9780262039000.

Design, development, integration: space shuttle primary flight software system. **Madden, William a Kyle, Rone.** New York: 1984, Communications of the ACM, str. 914–925. ISSN 0001-0782.

Does a Hybrid Approach of Agile and Plan-Driven Methods Work Better for IT System Development Projects? . **Imani, Takeomi, Nakano, Masaru a Anantatmula, Vittal.** 2017, International Journal of Engineering Research and Application, str. 39–46. ISSN 2248-9622.

DSDM: Dynamic Systems Development Method. **Stapleton, Jennifer.** Nancy: 1999. TOOLS Europe 1999: 29th International Conference on Technology of Object-Oriented Languages and Systems. ISBN 0-7695-0275-406.

Extreme programming and its development practices. **Juric, Radmila.** 2000, Proceedings of the 22nd International Conference on Information, str. 98–99. ISBN 1-59593-180-5.

Farcic, Viktor a Garcia, Alex. *Test-Driven Java Development, Second Edition: Invoke TDD principles for end-to-end application development, 2nd Edition.* Birmingham: Packt Publishing Ltd, 2018. str. 58. ISBN 978-1788832120.

Fowler, Martin. The New Methodology. *MartinFowler.com.* [online] 13. prosinec 2005. [citace: 9. září 2018] https://www.martinfowler.com/articles/newMethodology.html.

Fox, Steve. Behaviour-driven.org. *Behaviour-driven.org.* [online] 2. listopad 2016. [citace: 11. září 2018] http://behaviour-driven.org.

Framework for Agile Methods Classification. **Iacovelli, Adrian a Souveyet, Carine.** Paříž: 2008, Proceedings of MoDISE-EUS, str. 97. CEUR-WS.org.

Ganis, Matt. *Agile Methods: Fact or Fiction.* Hawthorne: International Business Machines, 2010. str. 3.

Garrido, Jose M. *Introduction to Computational Modeling Using C and Open-Source Tools.* Kennesaw: CRC Press, 2013. str. 11. ISBN 1482216795.

Goldman, Steven L. a Nagel, Roger N., Preiss, Kenneth. *Agile competitors and virtual organizations: Strategies for enriching the customer.* New York: Van Nostrand Reinhold, 1995. str. 4. ISBN 978-0471286509.

Grenning, James. Wingman-SW. *Planning Poker or How to avoid analysis paralysis while release planning.* [online] 4. dubna 2002. [citace: 11. března 2018] https://wingman-sw.com/papers/PlanningPoker-v1.1.pdf

Günal, Volkan. *Agile Software Development Approaches and Their History.* Bonn: University of Bonn, 2002. str. 4.

Highsmith, Jim. History: The Agile Manifesto. [online] 13. únor 2001. [citace: 1. říjen 2018] http://agilemanifesto.org/history.html.

Historical Roots of Agile Methods: Where did "Agile Thinking" come from? **Abbas, Noura, Gravell, Andrew a Wills, Gary.** Southampton: University of Southampton, 2008. Lecture Notes in Business Information Processing. str. 1. ISSN 1865-1348.

Inter-team coordination in large-scale globally distributed Scrum: Do Scrum-of-Scrums really work? **Paasivaara, Maria, Lassenius, Casper a Heikkilä, Ville T.** 2012, Empirical Software Engineering and Measurement (ESEM), str. 235–238. Esem-conferences.org

Is it all a game? Understanding the principles of gamification. **Robson, Karen, a další.** 2015, Business Horizons, str. 6. DOI 10.1016/j.bushor.2015.03.006.

Iterative and Incremental Development: A Brief History. **Larman, Craig a Basili, Victor R.** 2003, IEEE Computer, str. 47–48. ISSN 1558-0814.

Kassem, A. Saleh. *Software Engineering.* Fort Lauderdale: J. Ross Publishing, 2009. ISBN 1932159940.

Kekst, Brad. Stacey Matrix Mis-used Again! *Brad Kekst, Experienced Security Software Product Manager.* [online] 4. únor 2014. [citace: 3. únor 2019] http://bkekst.weebly.com/bk-blog/stacey-matrix-mis-used-again.

Laplante, Philip A. *What Every Engineer Should Know about Software Engineering.* Boca Raton: CRC Press, 2007. str. 28. ISBN 0849372283.

LEI. Muda, Mura, Muri. *Lean Enterprise Institute.* [online] 20. únor 2000. [citace: 5. únor 2019] https://www.lean.org/lexicon/muda-mura-muri.

Lester, Albert. *A guide to the Project Management Body of Knowledge (PMBOK guide) & Agile practice guide bundle.* Newtown Square: Project Management Institute, 2017. ISBN 978-1628251845.

Li, Jinjin. *Agile Software Development.* Berlín: Technische Universitt Berlin, 2012. str. 3.

Liker, Jeffrey. *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer.* New York: McGraw-Hill, 2004. ISBN 0071392319.

Managing the development of large software systems. **Royce, Winston Walker.** 1970, IEEE WESCON, str. 328–338. ISSN 2162-6634.

Martin, James. *Rapid Application Development.* Indianapolis: Macmillan, 1991. ISBN 0-02-376775-8.

Martinelli, Russ J. a Milosevic, Dragan Z. *Project Management ToolBox: Tools and Techniques for the Practicing Project Manager.* Hoboken: John Wiley & Sons, 2016. str. 318. ISBN 0471208221.

Matkovic, Pedja a Tumbas, Pere. A Comparative Overview of the Evolution of Software Development Models. *Journal of Industrial Engineering and Management.* 24. prosinec 2010, str. 166. ISSN 2013-0953.

Mixed agile/traditional project management methodology – reality or illusion?

Špundak, Mario. Záhřeb: 2014, 27th IPMA World Congress, str. 939–948.

No Silver Bullet: Essence and Accidents of Software Engineering. **Brooks, Frederick. 1986.** Los Alamitos: 1986, IEEE Computer, str. 13. ISSN 1558-0814.

North, Dan. *How to sell BDD to the business.* [online] 27. listopad 2009. [citace: 24. leden 2019] https://skillsmatter.com/skillscasts/923-how-to-sell-bdd-to-the-business

Ohno, Taiichi. *Toyota Production System – Beyond Large-scale Production.* Londýn: CRC Press, 1988. str. 176. ISBN 978-0-91529-914-0.

On the Impact of Kanban on Software Project Work An Empirical Case Study Investigation. **Ikonen, Marko, a další.** Engineering of Complex Computer Systems (ICECCS), 2011, IEEE, str. 2. ISBN 978-0769505831.

Palmer, Steve a Felsing, Mac. *A Practical Guide to Feature-Driven Development.* Upper Saddle River: Prentice Hall PTR, 2002. ISBN 0130676152.

Parekh, Nilesh. Aspects of the Waterfall Model Explained. *Techspirited.* [online] 4. listopad 2018. [citace: 24. leden 2019] https://techspirited.com/the-waterfall-model-explained.

Pichler, Roman. *Agile Product Management with Scrum: Creating Products that Customers Love (Addison-Wesley Signature Series (Cohn).* Boston: Addison-Wesley Professional, 2010. ISBN 0321605780.

Project Mercury: First step on the way to the moon. **Brownlee Jr., Henry T.** 2009, Boeing Frontiers, str. 9. Boeing.com/features/frontiers.

Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create.* New York: Crown Business, 2011. ISBN 978-0307887894.

Roock, Stefan. akquinet. *IT-agile.* [online] 1. duben 2007. [citace: 25. listopad 2018] https://www.it-agile.de/fileadmin/docs/FDD-Interview_en_final.pdf.

Roudias, Jihane. *Mastering principles and practices in PMBOK, Prince 2 and Scrum: Using Essential Project Management Methods to Deliver Effective and Efficient Projects.* Upper Saddle River: Pearson Education, 2015. str. 148. ISBN 0134060873.

Rzepecki, K. Graphic Products. *Value Stream Mapping (VSM).* [online] 2. ledna 2019. [citace: 15. února 2019] https://www.graphicproducts.com/articles/value-stream-mapping-vsm/.

Salnikov, Denis. Cross-functional teams and self-organization in the heart of Agile, part 2. *Medium.* [online] 28. říjen 2017. [citace: 14. únor 2019] https://medium.com/@d.salnikov23/cross-functional-teams-and-self-organization-in-the-heart-of-agile-part-2-7b2c6846d3bf.

Scrum Task Allocation Based on Particle Swarm Optimization. **Brezočnik, Lucija, Fister jr., Iztok a Podgorelec, Vili.** 2018, Bioinspired Optimization Methods and Their Applications, str. 38–49. ISBN 3319916408.

ScrumAlliance. Scrum Values. *ScrumAlliance.* [online] 5. leden 2019. [citace: 8. únor 2019] https://www.scrumalliance.org/learn-about-scrum/scrum-values.

Shewhart, Walter Andrew. *Statistical Method from the Viewpoint of Quality Control.* Washington, D.C.: Graduate School of the Department of Agriculture, 1939. str. 45.

Schwaber, Ken a Sutherland, Jeff. The Scrum Guide™. *Scrum.org.* [online] 3. březen 2017. [citace: 7. únor 2019] https://www.scrum.org/resources/scrum-guide.

Schwaber, Ken. *SCRUM Development Process.* Burlington: Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications, 1995. ISBN 0-89791-703-0.

Software Crisis 2.0. **Fitzgerald, Brian.** 2012, IEEE Computer, str. 89. ISSN 1558-0814.

Software Engineering Methodologies: A Review of the Waterfall Model and ObjectOriented Approach. **Adenowo, Adeokunbo A. A. a Adenowo, Basirat A.** 2013, International Journal of Scientific & Engineering Research, str. 429. ISSN 2229-5518.

Souza, Jamila Peripolli, Zavan, André Ricardo a Flôr, Daniela Eloise. *Scrum Hero: Gamifying the Scrum Framework.* Springer International Publishing, 2017. str. 131–135. ISBN 978-3-319-55907-0.

Stacey, Ralph. *Complexity and creativity in organizations.* San Francisco: Berrett-Koehler Publishers, 1996. ISBN 1881052893.

Stapel, Kai, Lübke, Daniel a Knauss, Eric. *Best Practices in eXtreme Programming Course Design.* Hannover: Leibniz Universität, 2008. ISBN 978-1-60558-079-1.

Stephens, Matt a Rosenberg, Doug. *Testování softwaru řízené návrhem.* Praha: Computer Press, Albatros Media a.s., 2017. str. 191. ISBN 8025145557.

Suzaki, Kioyshi. *New Manufacturing Challenge: Techniques for Continuous Improvement.* New York: Simon & Schuster, 1987. str. 8. ISBN 0029320402.

Takeuchi, Hirotaka a Nonaka, Ikujiro. *The New New Product Development Game.* Cambridge: Harvard Business Review, str. 137-146. 1986.

Thangarajoo, Yagulawathi. *Lean Thinking: An Overview. Industrial Engineering and Management.* Victoria: University of Melbourne, 2015. str. 2–4. DOI 10.4172/2169-0316.1000159.

The Evolution of Software Process Models: From the Waterfall Model to the Unified Modelling Language (UML). **Subair, Saad.** 2014, International Journal of Information Technology & Systems, str. 8. DOI 10.4018/IJITSA.

The waterfall model in large-scale development. **Petersen, Kai, Wohlin, Claes a Baca, Dejan.** Karlskrona: Springer, 2009. International Conference on Product-Focused Software Process Improvement. str. 386–400. ISBN 978-3-642-02152-7.

Todd, Sedano, Ralph, Paul a Péraire, Cécile. *The Product Backlog.* Montréal: International Conference on Software Engineering, 2019. ICSE-conferences.org.

Verma, Amit. Top 40 Agile Scrum Interview Questions (Updated). *WhizLabs.* [online] 30. květen 2017. [citace: 3. únor 2019] https://www.whizlabs.com/blog/agile-scrum-interview-questions/.

W Edwards Deming: Father of quality management, patient and composer. **Best, Mark a Neuhauser, Duncan.** 2005, Quality and Safety in Health Care, str. 310. ISSN: 1475-3901.

Wagenaar, Gerard, a další. *Artefacts in Agile Software Development.* Utrecht: Utrecht University, 2015. Lecture Notes in Computer Science. ISSN 0302-9743.

Waste analysis in self-service process. **Noya, Sunday a Chandra, Wie Wie.** 2014, Jurnal Ilmiah Teknik Industri, str. 168. ISSN 2460-4038.

Williams, Laurie a Kessler, Robert. *Pair Programming Illuminated.* Boston: Addison-Wesley Longman Publishing Co., Inc., 2002. str. 3. ISBN 978-0201745764.

Womack, James P. a Jones, Daniel T. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation.* Simon & Schuster, 1996. ISBN 0743249275.

Xiong, Jay. *New Software Engineering Paradigm Based on Complexity Science: An Introduction to NSE.* Londýn: Springer Science & Business Media, 2011. str. 32. ISBN 1441973265.

Yip, Jason. MartinFowler.com. *It's Not Just Standing Up: Patterns for Daily Standup Meetings.* [online] 21. únor 2016. [citace: 9. září 2018] https://www.martinfowler.com/articles/itsNotJustStandingUp.html.

Příloha č. 1 – výchozí seznam otázek k rozhovoru

1. Prosím o představení projektu Ω.
    a) O jaký software se jednalo? Komu měl být dodán?
    b) Jaká byla odhadovaná priorita projektu ve srovnání s ostatními projekty?
2. Byla organizační struktura podniku přizpůsobena agilnímu řízení?
3. Jaká byla angažovanost top managementu ve vztahu k agilním metodikám; Scrum týmu?
4. Bylo v rámci organizace více Scrum týmů?
5. Které role byly rozeznávány ve Scrum týmu? Kdo patřil do týmu vývojářů?
6. Jak byly rozděleny odpovědnosti v rámci týmu?
7. Jaké schůzky se v rámci projektu pořádaly?
    a) Jaká byla přibližná účast?
    b) Koncentroval se tým plně na obsah těchto mítinků?
8. Jak dlouhé byly sprinty, resp. frekvence dodávek?
9. Dalo by smýšlení týmu označit za agilní?
10. Jakým způsobem reagoval tým na nové požadavky zákazníka?
11. Dá se o daném týmu říct, že se sám řídil?
12. Napomínal jste někoho jako Scrum master? V jakém ohledu?
13. Jak byste zhodnotil stav morálky týmu? Byl motivován k dosažení výsledku?
14. V rámci neformálních vztahů – o čem jste se bavili?
    a) Bavili jste se o hrách?
    b) Jaký žánr byl nejvíc zmiňován?
15. Shrnete pozitivní a negativní stránky týmu z perspektivy Scrum Mastera?
16. Jakou metodikou byl projekt reálně veden?

Příloha č. 2 – přepis rozhovoru

T: = tazatel
R: = respondent

T: Dobrý den, jsem tu proto, abych provedl rozhovor s respondentem J. W. ohledně diplomové práce na téma Gamifikace jako technika k zefektivnění Scrumu ve vybrané společnosti. Chtěl bych ho tímto přivítat: dobrý den.

R: Dobrý den.

T: První otázka se bude týkat samotného projektu Ω – jeho představení, o jaký software se jednalo, komu měl být dodán a jaká byla priorita daného projektu?

R: Jednalo se o webovou aplikaci, která byla dodávána pro jednu nejmenovanou velkou americkou banku. Měla sloužit jako administrativně-reklamní nástroj pro designování kreditních karet, kdy v Americe jsou kreditní karty velký byznys a podle různých průzkumů, když je kreditní karta personalizovaná, tak je preferována danými uživateli, a jelikož v Americe každý nebo většina lidí má více kreditních karet, protože na každé kreditní kartě jsou třeba různé slevy a různé promo akce, tak podle těch daných průzkumů se ukázalo, že když je právě taková kreditní karta personalizovaná, tak jí bude spíše daný uživatel té karty používat s tím, že tento portál byl zaměřený na středně velké podniky: klienty dané banky – většinou různé profesní organizace, kterým byl nabídnut jako služba; ony si mohly vytvořit v rámci předdefinovaných parametrů svůj vlastní design té kreditní karty pro své členy. Řekněme např. asociace v rámci jedné univerzity, tak ta univerzita si vlastně odebrala tu aplikaci, vytvořila si design karet např. se svým logem, nějakým heslem, ale k tomu bylo podložený třeba ještě logo té banky, a pak to nabízela svým členům. Výhoda byla v tom, že bance se zvýšil obrat používáním kreditních karet a daná organizace z toho měla nějaký podíl na zisku z těch plateb a z používání těch kreditních karet, takže to byl oboustranně výhodný obchod pro takové větší klienty banky, kteří potom dále takhle zprostředkovávali ony kreditní karty.

T: V rámci portfolia projektu to byl, řekněme, větší projekt?

R: Byl to projekt, nevím, jak přesně definovat, jako větší bych úplně neřekl – to si představuji jiné projekty, ale byl to projekt, na kterém pracovalo mezi šesti až deseti vývojáři, a projekt trval vlastně přes rok a půl. Takže spíše středně velký IT projekt.

T: Jednalo se tedy o Scrum tým. Jaké role byly rozeznávané v rámci toho Scrum týmu? Pokud tam byl definovaný tým vývojářů, kdo spadal do týmu vývojářů?

R: Scrum tým to víceméně byl, ale s tím, že nebyly striktně dodrženy role, minimálně ze začátku, definované ve Scrumu. Začínali jsme rozděleni jako vývojový tým. Za vývojový tým se považovali jak developeři, tak testeři, ale obecně byli označeni jako development tým, což víceméně odpovídalo Scrum metodice, ale například ze začátku jsme měli projekt manažery – nebyli product owneři. Do té role product ownerů se postupně projekt manažeři vlastně přeměňovali v rámci toho projektu, protože ten projekt byl takovým přechodným projektem v rámci přechodu z metodik víceméně částečně Waterfall a částečně žádných metodik na metodiku Scrumu.

T: A jak v takovém týmu byly rozděleny třeba odpovědnosti?

R: Tam se naráželo na největší problém, co se týče Scrumu, a to je především zodpovědnost za práci a přidělování práce jednotlivým lidem, kdy ta zodpovědnost neležela na vývojovém týmu, ale ležela buď na product ownerech, nebo project manažerech ze začátku a částečně na line manažerech jednotlivých vývojářů.

T: Jaký typ schůzek se u vás konal v rámci toho Scrumu? Jaká byla účast? Dál se budu ještě doptávat.

R: V rámci Scrumu byly schůzky paradoxně jedna z mála věcí, které víceméně Scrumu odpovídaly skoro přesně. Konaly se každodenní stand-upy, na kterých se účastnil vývojový tým a většinou byl přítomen i product owner s tím, že se tam řešila standardní agenda. Víceméně každý řekl, na čem v současnosti pracuje a zdali má nějaké problémy s prací nebo ne. Byly pravidelné retrospektivy, kterých se účastnil celý tým včetně line manažerů a product ownerů. Retrospektiva byla používána nejenom k řešení problémů nebo toho, jak se říká ve Scrumu, co se nám povedlo v předchozím sprintu, ale také si tam většinou mohli vývojáři představit svoji práci, co za ten sprint udělali, a je to kvůli tomu, že bohužel se ten projekt dodával velké bance a banka nebyla ochotna přistoupit na pravidelnou prezentaci práce vývojářů po každém sprintu, takže se prezentovala jednou za dva až tři měsíce.

T: Došlo tedy ke spojení zhodnocení sprintu a zároveň těch retrospektiv?

R: Víceméně ano, ale s tím že nebyl přítomen klient ani na jednom.

T: Dobře.

R: Ještě se vlastně dělal grooming jednou týdně a tam většinou seděli line manažeři, product owneři a občas senior vývojáři, kdy se upravovaly odhady jednotlivých tasků a upřesňovaly požadavky, a tak podobně. Což znovu částečně odpovídá té metodice. Neodpovídá to z toho hlediska, že by tam měli především sedět vývojáři, ale nebyla v rámci firmy ochota na přenos této zodpovědnosti.

T: Jak jste sledoval třeba koncentraci těch lidí na těch jednotlivých mítincích? Byli koncentrovaní nebo bylo nějaké rozptýlení? Jde mi o tu jejich pozornost, té věnované problematice, jestli se zkrátka nevěnovali jiné aktivitě.

R: Stand-upy většinou ano, protože tam jsme se snažili dodržovat těch deset a patnáct minut, takže tam nebyl problém s udržením pozornosti. Co se týče groomingu, tam to bylo komplikovanější, ale většina těch účastníků se na to soustředila, protože to byl docela důležitý meeting. A co se týče retrospektiv, tak tam to hodně záleželo, protože když už se ta retrospektiva protahovala, tak tam bylo vidět, že vývojáři, kteří v tu danou chvíli nemluví nebo členi týmu v tu danou chvíli nemluví, tak přesně, třeba v tu chvíli sledovali telefon nebo dělali něco na notebooku, věnovali se něčemu jinému.

T: Mě by také zajímalo, jak dlouhé byly sprinty; jaká byla frekvence dodávek?

R: Sprinty jsme měli dvoutýdenní s tím, že teoreticky končily víceméně tím, že byl nějaký inkrement, ale nefungovalo to úplně na principu dodávek. Jak jsem říkal: klient byla banka, která spíše jela ve Waterfallové struktuře, takže vyloženě klientovi se dodávalo ve třech fázích za celý ten projekt. Vlastně za rok a čtvrt, a před kterým předcházel velký regresní testing z hlediska našeho týmu a potom probíhal komplexní UAT testing na straně klienta. Pak byla vlastně dodávka nějaké fáze 1, 2, 3 daného projektu.

T: Když se zaměříme na ten tým jako takový, dalo by se to smýšlení toho týmu označit za agilní? Z vaší perspektivy?

R: Víceméně by se to dalo tak pojmout, protože ten tým byl tým lidí, kteří se znali dlouhodobě. dlouhodobě spolupracovali, ta zodpovědnost za vlastní práci tam byla na velmi vysoké úrovni. Mezi sebou si ti členi důvěřovali a co je velmi důležité, tak ta mentalita byla Scrumová. Prostě když byl nějaký velký problém, tak se nikdy neřešilo, kdo ten problém způsobil, naopak se spíše přihlásili tři různí lidé, kteří řekli: „Jo, je to způsobený mnou…“ nebo „Já jsem tomu přispěl.“ Ta zodpovědnost vždycky ležela na celém tom týmu, který to

tak bral, že to je jejich zodpovědnost, a ne zodpovědnost nějakého jedince, který udělal něco špatně.

T: Ještě bych se zaměřil na zákazníka, protože by mě zajímalo, jak tam probíhalo to, že vlastně hodnotou toho Scrumu je, že ten zákazník nějakým způsobem mění požadavky. Jak to tedy fungovalo?

R: No tohle nefungovalo úplně tak, jak by Scrum fungovat měl, spíš to fungovalo špatně než dobře z tohoto hlediska. Byl problém s tím, že se přesně zákazník na to dva měsíce pořádně nepodíval, potom se podíval na to, co se dodávalo. Půlka věcí se mu tam nelíbila a potom se to narychlo měnilo, což je přesně to, čemu by ten Scrum měl předejít a potom se to bohužel řešilo vytvářením byznys analýzy a test analýzy, která fungovala jako akceptační kritéria a potom tam byl takový přeliv vlastně toho Waterfallového přístupu, co se týče toho klienta, ale snaha z naší strany byla, aby klient pravidelně připomínkoval, co se mu líbí, co ne ale, bohužel ta spolupráce s klientem tam z tohoto hlediska byla špatná.

T: Tudíž byste ocenili větší participaci zákazníka v rámci toho projektu.

R: Ano, aby lépe ten projekt fungoval z hlediska Scrum metodiky, tak rozhodně. Neřekl bych větší, ale spíš frekventovanější participaci. On nebyl jakoby problém rozsah té participace, ale problém byl v tom, že zákazník prostě na tom participoval jednou za velmi dlouhý časový úsek, a potom tomu věnoval klidně tři týdny čistého času. Spíš by bylo lepší, kdyby ten zákazník tomu věnoval jeden, dva, tři dny každý sprint, než aby tak rigorózně k tomu přistupoval tou Waterfallovou metodou, jako „My ten projekt převezmeme a pak vám k tomu dáme připomínky.“

T: Obzvláště, když ty sprinty trvají dva týdny… Já bych se chtěl ještě zeptat na to, jestli se tedy o tom týmu, když tedy bychom označili to, že svým způsobem agilní byl, jestli byl i samořiditelný, jestli se ty lidi dokázali sami řídit, sami si organizovat práci a sami vytvářet nějaké otázky a úkoly i mimo to, že jim samozřejmě nějaké navrhuje product owner?

R: No, tohle to je další věc. Tady byla další komplikace v rámci toho týmu, že lehce bokem toho Scrumu fungoval stále line management, který částečně do té organizace zasahoval. Co se týče fungování, ten tým byl dobrý v tom, že oni opravdu byli ochotni za tu svoji práci zodpovídat, a když něco nevěděli, něco se jim nezdálo nebo neměli co dělat, tak se vždycky ozvali a snažili se to řešit, ale stále to bylo v relativně rigorózním korporátu, který zasahoval do té zodpovědnosti za jednotlivé věci. Tudíž, co se týče třeba vytváření úkolů a plánování

nebo asignování těch úkolů, tak tohle většinou bohužel probíhalo z vrchu – tedy z pozice project manažerů nebo line managementu.

T: Jinak nebylo potřeba, abyste, dejme tomu, někoho napomínal?

R: A tak, jako občas se napomínalo. Napomínali se paradoxně nejvíce product owneři, kteří občas zbytečně moc zasahovali právě do té práce. Spíše ale moje pozice byla o tom, že jsem vysvětloval, napovídal, trochu podporoval ty vývojáře ve změnách, snažil jsem se jim vysvětlit, v čem je ten přístup lepší, i když třeba ze začátku se jim to zdálo nesmyslné – některé postupy, ale bylo to spíše na té pozici jako poradit, vysvětlit, než že by se napomínal někdo.

T: A co zhodnotit tu morálku toho týmu, jak byl ten tým motivován k dosažení toho výsledku?

R: Na tomhle projektu zrovna byla ta morálka dobrá, protože ten projekt byl něco nového pro hodně členů toho týmu. Pracovali na nejmodernějších technologiích v rámci toho týmu, takže oni si to docela užívali z toho hlediska, že se učí nové věci, dělají něco jiného, než dělali do teď a vlastně tak trochu byli motivovaní tou zodpovědností za tu práci, že byli hrdí na to, co dělají a co dodávají, že to je zajímavé a že to hezky vypadá a tak... I když chyběla taková ta motivace, že to představí klientům a jsou hrdí na to, že se to těm klientům líbí, tak tam probíhalo to představování svojí práce zbytku týmu – a teď nemyslím Scrum týmu, ale komplet organizaci, která byla výrazně širší než Scrum tým. Bylo to třeba i potom vidět ke konci toho projektu, kdy se ten projekt dostal lehce do skluzu a probíhaly potom crunche, což jsou v podstatě přesčasy, kdy se vyvíjí. Skutečně, třeba dva, tři týdny se chodilo na deseti nebo dvanácti hodinové směny, ale ty byly absolutně nenařízené. Prostě ten tým sám věděl, že jsou takové a takové problémy a zkrátka strávili v práci více času, aby se projekt dodal v požadovaném stavu a požadovaném čase.

T: Takže jste spolu vycházeli i v rámci neformálních vztahů?

R: Ano, jako ten tým v rámci neformálních vztahů fungoval velice dobře. Já jsem tam byl v podstatě outsider, protože jsem do toho týmu nastupoval poslední, ale jak jsem říkal, tak většina lidí nebo většina členů týmu, to byli lidé, kteří spolu dlouhodobě pracovali. Většina okolo tří, čtyř let, a hlavně to byl původně tým, který patřil startupu, a hodně lidí se znalo ještě z doby startupu, takže to byli lidi, kteří spolu trávili i volný čas a moc se to nezměnilo

ani ve chvíli, kdy byl ten startup koupený velkým korporátem a já jsem nastupoval v době toho korporátu, ale ty dobré vztahy z éry startupu tam zůstávaly.

T: O čem jste se takhle bavili v rámci těch neformálních vztahů? Co jste probírali jako za své zájmy.

R: To záleží na tom, kdo měl jaké zájmy. Bavili jsme se někdy i o pracovních věcech, ale ne jako přímo o projektu, ale třeba o technologiích, které s tou naší prací souvisejí, protože to hodně těch lidí zajímalo, bavili jsme se o autech, o vztazích... jako přátelé v podstatě.

T: Bavili jste se třeba i o hrách?

R: I o hrách jsme se bavili, i když to jsem se bavil především s jedním kolegou, který rád hrál, hrál rád i jeho syn, ale s dvěma nebo třemi ostatními kolegy, se kterými jsem často přicházel do styku, jsme se o hrách nebavili z toho důvodu, že oni hry už na konzolích, ani na počítačích moc nehráli v tu dobu.

T: Takže tam ani nebyl nějaký vymezený řekněme žánr vyloženě?

R: Ne, rozhodně žádný vymezený žánr nebyl. To byla prostě taková běžná přátelská debata.

T: Mohl byste shrnout všechny pozitivní a negativní stránky týmu, který jste tak nějak zmínil, a třeba i nějaké přidal?

R: Z té pozitivní stránky určitě ten dobrý neformální vztah mezi jednotlivými členy je pozitivní stránka, zodpovědnost za svoji práci je také rozhodně pozitivní stránka. Byli tam také schopní vývojáři, což je taky velmi pozitivní stránka a je to důležité pro projekty. Co se týče negativních, někdy dobré neformální vztahy můžou mít negativní vliv na projekty, protože občas line manažeři dovolí věc, kterou by běžně nedovolili. Nejvíc negativní nebylo stejně ale spojené s tím týmem, ale bylo to dané typem projektu – spíše typem klienta a typem společnosti, ve které jsme pracovali, a to byla vlastně ta strukturovanost korporátu – zažitá nějaké pravidla a zodpovědnosti a tam byl velký problém s přenosem zodpovědnosti tak, jak by měl probíhat ve Scrumu, a to nebyl prostě problém týmu, ale problém organizace, pro kterou jsme pracovali.

T: Takže ta organizace nebyla přizpůsobená agilnímu řízení?

R: Ne, z mé zkušenost i velké korporáty se málokdy přizpůsobují na agilní řízení, protože je tam striktně nastavené, kdo za co zodpovídá a většinou jsou to lidé výše postavení než to, co by předpokládal Scrum.

T: Jak se třeba angažoval top management ve vztahu k agilnímu řízení nebo k tomu vašemu týmu?

R: To já úplně nemohu hodnotit z toho hlediska, že já jsem se s top managementem nesetkal. Nedostal jsem se s ním do styku z toho důvodu, že společnost, ve které jsem pracoval, byla dosti hierarchizovaná a vlastně šéf celého našeho týmu, který měl pod sebou pětadvacet lidí, a teď myslím tým organizačně, ne tým Scrumově, tak sám se zodpovídal řediteli, který byl stále ještě tři úrovně pod top managementem. Takže tam ta strukturovanost té firmy byla obrovská, tam těch úrovní bylo opravdu hodně.

T: V rámci té organizace bylo více Scrum týmů nebo jste byli jediní?

R: Bylo tam více Scrum týmů, byly v úplně jiných odděleních, s nimi jsme neměli nic společného, takže já úplně nemůžu hodnotit jak fungovali, ale třeba tým se kterým jsme spadali do jedné sekce, u kterého jsem občas slyšel o sdílení nějakých technologií a metodik a podobně, tak popravdě, přestože si říkali Scrum tým a měli tří týdenní sprinty, tak co já jsem vypozoroval, tak to vůbec nebyl Scrum, ale byl to spíše způsob vývoje mini-Waterfallu, kde si rozdělili ten obrovský projekt na velmi mnoho malých Waterfallů na tři týdny.

T: Takže jste nesdíleli nějaké zkušenosti jako Scrum masteři?

R: Ne, to jsme nesdíleli bohužel vůbec. Co se týče sdílení zkušeností, tak probíhalo možná částečně na úrovni technických znalostí mezi vývojáři v rámci společnosti nebo organizace, ale mimo sdílení těchto znalostí tam žádné sdílení neprobíhalo. Popravdě bych si skoro dovolil tvrdit, že v rámci kanceláře pražské pobočky jsme byli pouze dva Scrum masteři, protože třeba ten tým, o kterém jsem mluvil, tak byl ve Francii.

T: Takže se nedá mluvit o nějakém Scrum of Scrums? Neodpovídalo to vůbec organizační struktuře vůbec společnosti, ale zároveň tam nebyla spolupráce?

R: Ne, vůbec.

T: No, a kdybychom měli celkově zhodnotit projekt Ω, tak jakou metodikou byl reálně veden, byl to Scrum nebo co to bylo?

R: No, tohle jako těžko hodnotit. Snaha byla se Scrumu co nejvíce přiblížit, to je hodně důležité. Já bych řekl, že nebyl řízený Scrumem, ale ani žádnou jinou metodikou. Důležité je, že opravdu byla velká snaha a podpora včetně line manažerů se Scrumu co nejvíce přiblížit, ale z druhé strany se taky vědělo například, že ne úplně všechno je v rámci té naší organizace reálné, například přenos těch zodpovědností na vývojáře, a proto se muselo postupovat po malých krůčkách, dělat malé úpravy a postupně zavádět různé prvky z té Scrumové metodiky. Takže já říkám, že asi jsme byli jeden z nejvíce Scrum týmů v rámci společnosti nebo jako nejvíc jsme se Scrumu blížili, ale čistý Scrum to prostě nemohl být, protože to nebylo v rámci toho korporátu možné.

T: Jak nakonec dopadl projekt?

R: Projekt byl zdárně dodán klientovi i přes to, že ve finální fázi tam byly problémy, ale to jsem říkal. Prostě se tým do toho vrhl po hlavě, problémy vyřešil a ta webová aplikace je v dnešní době klientem už nějakou dobu používaná.

T: Myslíte takhle retrospektivně, že by tam mohl být nějaký prostor pro zlepšení a užití nějaké dodatečné metodiky nebo jakou to přineslo zkušenost?

R: No já si myslím, že prostoru pro zlepšení by tam bylo hodně, ale tak jako to je na velký části projektu. Důležité je se z toho poučit, třeba jedna z největších nebo pro mě z mého pohledu jako Scrum mastera dvě věci, co já bych tam nejraději zlepšil by bylo první ta komunikace s klientem, respektive jeho pravidelný feedback, aby se potom nestalo, že najednou při dodávce si klient vzpomene, že polovinu věcí chtěl jinak než dodáváme a potom se to muselo měnit a druhá věc je přenos více zodpovědností na tým s tím, že ideálně taky neplánovat tak přesně s časem, to byl podle mě jeden z problémů nebo co se mi úplně nelíbilo, i když ta možnost v rámci metodiky Scrumové operovat s časem je, tak já dávám přednost story pointům. Tady byl velmi rigorózní přístup k tomu časovému plánování, a to z důvodu účtování práce ke klientovi, a to je další problém tady zrovna tohoto korporátu, že se klientům musí naúčtovat přesně strávené hodiny na tom, co přesně ti vývojáři a členové týmu dělají a potom to vede k tomu, že jednotlivé story, otázky, kroky a bugy jsou oceňované velmi rigorózně časem, který potom ne vždy odpovídá realitě, protože to jde těžko odhadnout.

T: Dobře, za mě je to všechno. Děkuji za poskytnutí rozhovoru a na shledanou.

R: Není zač, na shledanou.