TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Agilní metody v projektovém řízení

Diplomová práce

Studijní program: N6209 – Systémové inženýrství a informatika
Studijní obor: 6209T021 – Manažerská informatika

Autor práce: **Bc. Petr Halama**
Vedoucí práce: doc. Ing. Klára Antlová, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Economics

Agile methods in project management

Diploma thesis

Study programme:	N6209 – System Engineering and Informatics
Study branch:	6209T021 – Managerial Informatics
Author:	**Bc. Petr Halama**
Supervisor:	doc. Ing. Klára Antlová, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Halama**
Osobní číslo:	**E14000389**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Manažerská informatika**
Název tématu:	**Agilní metody v projektovém řízení**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

1. Principy a trendy projektového řízení
2. Porovnání agilních a klasických metod (výhody a nevýhody)
3. Průzkum využití agilního přístupu pomocí případové studie
4. Návrh zavedení agilního přístupu

Rozsah grafických prací:

Rozsah pracovní zprávy: **65 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

ŠOCHOVÁ, Zuzana a Eduard KUNCE. Agilní metody řízení projektů. Brno: Computer Press, 2014. ISBN 978-80-251-4194-6.
PROCHÁZKA, Jaroslav a Cyril KLIMEŠ. Provozujte IT jinak: agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb. Praha: Grada Publishing, 2011. ISBN 978-80-247-4137-6.
UNHELKAR, Bhuvan. The art of agile practice: a composite approach for projects and organizations. Boca Raton: CRC Press, 2013. ISBN 9781439851180.
Elektronická databáze článků ProQuest (knihovna.tul.cz).

Vedoucí diplomové práce: **doc. Ing. Klára Antlová, Ph.D.**
Katedra informatiky

Konzultant diplomové práce: **Ing. Lukáš Voplakal**
UVM interactive, s.r.o.

Datum zadání diplomové práce: **31. října 2015**

Termín odevzdání diplomové práce: **31. května 2017**

doc. Ing. Miroslav Žižka, Ph.D.
děkan

L.S.

doc. Ing. Jan Skrbek, Dr.
vedoucí katedry

V Liberci dne 31. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych tímto poděkoval vedoucí mé diplomové práce doc. Ing. Kláře Antlové, Ph.D za ochotu, čas a věcné připomínky, které mi během konzultací poskytla. Dále bych chtěl poděkovat všem přátelům a rodině za jejich podporu a věcné rady.

Anotace

Diplomová práce se zabývá problematikou projektového řízení, konkrétně pak tradičními a především agilními metodikami. Teoretickou rešerši lze rozdělit na čtyři dílčí části, přičemž během úvodu jsou představeny základní pojmy a obecný pohled na problematiku různých přístupů k projektovému řízení. Poté se práce věnuje tradičnímu přístupu k projektovému řízení a podrobněji popisuje tři konkrétní tradiční metodiky. Následující a zároveň nejrozsáhlejší část teoretické rešerše tvoří sekce věnovaná agilnímu přístupu. V této části jsou podrobně popsány metodiky Scrum, Lean Development, DSDM a Extrémní programování. Na závěr teoretické části jsou oba přístupy zevrubně porovnány. Praktická část se pak věnuje analýze projektového řízení v rámci konkrétní organizace a návrhu alternativního agilního přístupu s uplatněním metodiky Scrum.

Klíčová slova

Projektové řízení, Tradiční metodiky, Vodopádový model, Spirálový model, Rational Unified Process, Agilní metodiky, Scrum, Lean Development, Extrémní programování, Dynamic Systems Development Method

Abstract

This thesis investigates the issue of project management; in particular it deeply explores agile methodologies and agile project management generally, however brief introduction of traditional project management and its methodologies are included as well. Following a brief general introduction into the topic, there is an overview given with the emphasis on different project management approaches. Furthermore the thesis provides a closer look on three methodologies that belong in the traditional project management approach. Subsequently the main issue of agile project management approach is investigated. As a result agile methodologies such as Scrum, Lean Development, DSDM and Extreme Programming are thoroughly introduced and summarized. The theoretical part is then concluded with comparison of the two project management approaches. Finally there is a case study that consists of analysis of a real project management environment in a specific company and a proposal of an alternative solution using the Scrum methodology which would tackle the imperfections of the previously used approach.

Key words

Project Management, Traditional methodologies, Waterfall model, Spiral model, Rational Unified Process, Agile methodologies, Scrum, Lean Development, Extreme Programming, Dynamic Systems Development Method

Obsah

Seznam obrázků a ilustrací

Seznam tabulek

Seznam použitých zkratek

APM	Agile Project Management
DSDM	Dynamic Systems Development Method
XP	Extreme Programming
xPM	Extreme Project Management
FDD	Feature Driven Development
IPMA	International Project Management Association
MPx	Emertxe Project Management
PMI	Project Management Institute
RAD	Rapid Application Development
RUP	Rational Unified Process
TDD	Test Driven Development
TPM	Traditional Project Management
UML	Unified Modelling Language

Úvod

Projekt a projektové řízení jsou pojmy, které už nějakou dobu frekventovaně zaznívají v podnikatelské sféře, ve školství, ve vládě a v mnoha dalších typech různých institucí. Jinými slovy, soukromý i veřejný sektor začaly stále častěji využívat formu projektu k realizaci svých záměrů, což zákonitě vedlo i k rozvoji projektového řízení jako samostatné disciplíny. Rozvoj celé disciplíny s sebou přinesl i vznik odlišných pohledů na celou problematiku, jež měly za následek vyprofilování různých přístupů, jak by měly být projekty řízeny a jaké principy jsou z hlediska úspěchu projektu klíčové. V dnešní době jsou pak rozlišovány především tradiční a agilní přístup k projektovému řízení, kterými se podrobněji zabývá i tato diplomová práce.

Práce je strukturovaná na dvě části, kdy první část je věnována teoretickému výkladu na základě literární rešerše, která je zároveň součástí tohoto výkladu. Po teoretické části následuje část praktická, jež je věnována případové studii, která podrobně analyzuje konkrétní projektové řízení na základě ilustračních projektů realizovaných v reálné organizaci.

Teoretická část poskytuje stručné představení tradičního a agilního přístupu a detailněji seznamuje čtenáře s konkrétními metodikami, které jim náleží. U metodik, které to umožňují, je nahlíženo na jejich uplatnění v obecnějším smyslu a práce usiluje o popsání jejich možné aplikace i v jiných oblastech mimo oblast softwarového vývoje, kterou se většina projektových metodik primárně zabývá. Důraz je pak kladen především na metodiky agilní, jelikož právě ty nabízejí efektivní řešení rostoucí dynamiky dnešního světa a lépe se potýkají s proměnlivými podmínkami, jež doprovází velké množství současných projektů. Nicméně práce se z části věnuje i problematice tradičních metodik, jelikož právě tradiční metodiky a jejich úskalí daly prvotní impuls k vytvoření prvních agilních metodik a Agilního manifestu softwarového vývoje, jenž je považován za základní dogma celého agilního smýšlení. Teoretická část této diplomové práce je následně završena podrobným srovnáním obou přístupů.

Praktickou část tvoří případová studie, jež se věnuje analýze podnikové metodiky, která byla v minulosti úspěšně aplikována k řízení interních projektů v rámci existující organizace. I přes úspěšnost projektů řízených za pomoci dané metodiky obsahoval zvolený přístup určitá úskalí, která dávají nemalý prostor k optimalizaci. Záměrem této části kvalifikační práce je pak ona úskalí eliminovat skrze návrh alternativy v podobě agilního přístupu s uplatněním některé z agilních metodik, kterým je věnovaná teoretická rešerše.

Cílem této diplomové práce je v rámci teoretické části seznámit čtenáře s problematikou tradičního a agilního přístupu, konkrétními metodikami, které jim náleží, a na závěr poskytnout zevrubné srovnání obou těchto přístupů. Praktická část si následně klade za cíl poskytnout návrh agilní alternativy na základě analýzy z případové studie a zhodnotit přínosy plynoucí z navrhovaného řešení.

1 Projekt a Projektové řízení

Projekt a s ním spojené projektové řízení jsou pojmy, které v dnešní době velmi často figurují ve formálních dokumentech, kde určují formu, již zaujmou časté zakázky soukromého i veřejného sektoru. Je tedy vhodné si oba pojmy stručně představit, což bude náplní této krátké kapitoly.

1.1 Projekt

Obecně lze projekt charakterizovat jako soustavu vzájemně propojených procesů a činností, jejichž cílem je vytvoření či změna konkrétního statku nebo služby, přičemž termín začátku i konce projektu jsou pevně stanoveny, stejně jako jeho rozsah či finanční a výrobní zdroje. Výstupem dokončeného projektu by mělo být něco unikátního [1], ve smyslu že práce na dosažení toho výstupu nejsou rutinní záležitostí, ale vyžadují koordinovanou práci projektového týmu, jež tvoří specialisté nezbytní k dosažení onoho výsledku. Příkladem takového projektu může být tvorba nového softwaru, stavba mostu či obytného domu, stejně jako expandování společnosti na nové trhy atd. Pro úplnost si zde uvedeme úplnou definici podle normy ISO 10006 [2], která projekt popisuje následujícím způsobem:

„Projekt je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji."

1.2 Projektové řízení

Přirozenou součástí integrace projektů do podnikových činností byl i vývoj projektového řízení. Obdobně jako u jakéhokoliv řízení má i projektové řízení za úkol zajistit efektivitu činností a procesů, navíc si však dává za cíl vyhovět požadavkům tzv. Projektového trojimperativu, někdy také nazývaného Magickým trojúhelníkem projektového řízení [3]. Jinými slovy, projekt se považuje za úspěšný za předpokladu, že bude dokončen a předán v termínu, v požadovaném rozsahu a s náklady nepřesahujícími požadovanou úroveň.

Projektovému řízení jako samostatné disciplíně se věnují na mezinárodní úrovni různé organizace jako třeba Project Management Institute (PMI), u nás známý spíše pod názvem Společnost pro projektové řízení, nebo International Project Management Association

(IPMA), které vytváří standardy projektového řízení či zastřešují certifikace pro nové projektové manažery [4] [5]. Ze standardů stojí za zmínku např. PMBOK, jenž je produktem společnosti PMI, a nebo PRINCE2 od společnosti Axelos.

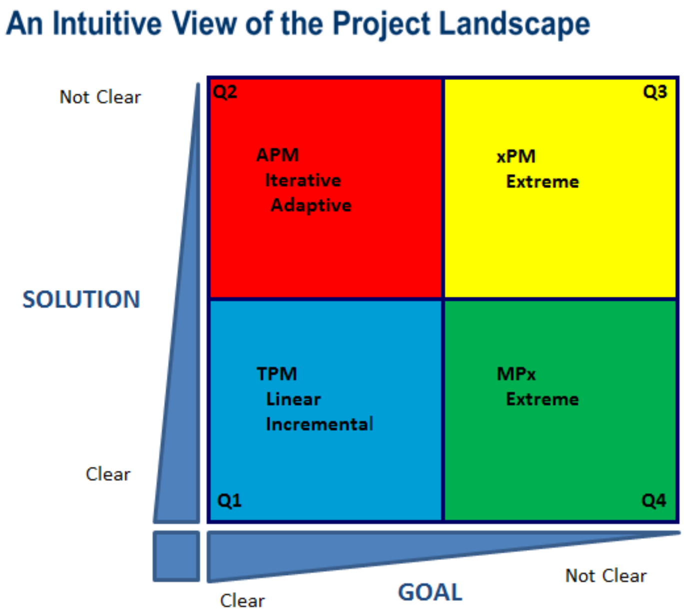
Navzdory existenci množství standardů, které si dávají za cíl projektové řízení nějakým způsobem sjednotit, je každý projekt už podle samotné definice specifický a unikátní. Je tedy zřejmé, že nelze ke každému projektu přistupovat totožně, naopak je nezbytné přistupovat ke každému projektu individuálně a s ohledem na jeho charakter. Z hlediska odlišných zásad a principů pro řízení jednotlivých projektů dnes rozlišujeme různé přístupy k projektovému řízení, které si představíme v následující kapitole.

1.3 Přístupy k projektovému řízení

Definujeme-li si u každého projektu cíl a řešení jako dvě proměnné, na kterých závisí volba metodiky či životního cyklu projektu, pak můžeme podle Wysockého [6, s. 312] rozdělit projektové prostředí do čtyř kvadrantů, viz Obrázek 1. Na základě toho, zdali jsou cíl a řešení u projektů jasně specifikovány nebo jsou nejasné, rozlišujeme následující přístupy:

- **Tradiční přístup (TPM)** vychází z předpokladu, že cíl i cesta k jeho dosažní jsou jasné. Zpravidla se jedná o známé nebo méně komplexní projekty, kde nehrozí velké riziko nečekaného překvapení či jiné nenadálé události [7]. Tyto projekty jsou vždy velmi dobře naplánované a zdokumentované, což napomáhá snižovat celkové riziko výskytu nenadálých událostí. Na druhé straně však tyto projekty neposkytují mnoho prostoru ke změnám ze strany zadavatele. Příkladem může být budování nové pobočky fastfood řetězce.
- **Agilní přístup (APM)** je využíván v případě, že cíl (problém) je definován, avšak cesta k jeho dosažení (řešení) nikoliv. Obvykle to jsou komplexní projekty, které jsou dekomponovány na menší samostatné celky či iterace, na konci kterých je zákazníkovi inkrementálně dodávána určitá přidaná hodnota. Agilní přístup je úzce vázán na spolupráci projektového týmu se zákazníkem, který podává zpětnou vazbu na aktuální stav a koriguje budoucí vývoj projektu změnovými požadavky. Příkladem může být vývoj nového softwarového řešení nebo reengineering podnikových procesů.

- **Extrémní přístup (xPM)** je zaměřen na oblast vědy a vývoje. Jde o riskantní a dlouhodobé projekty bez jasně definovaného cíle i jeho řešení. Příkladem takového projektu může být třeba hledání léku na rakovinu [7].
- **Obrácený extrémní přístup (MPx)** je obdobou extrémního přístupu avšak s jedním zásadním rozdílem – celý projekt je převrácený. V tomto případě je známé řešení – nová technologie, ale není známá její aplikace v praxi či business uplatnění [8]. Příkladem může být cloudové řešení nebo nanotechnologie, pro které se hledá business uplatnění [7].



Obrázek 1: Čtyři kvadranty projektového řízení dle Wysockého
Zdroj: **[7]**

Z hlediska dnes využívaných projektových metodik lze říci, že nejčastěji se v praxi setkáme s tradičním a agilním přístupem. Právě tyto dva přístupy jsou detailněji popsány v následujících kapitolách spolu s metodikami, které jim náleží.

2 Tradiční přístup k projektovému řízení

Tradiční přístup k projektovému řízení, anglicky Traditional Project Management, vychází z předpokladu, že zákazník je schopen identifikovat veškeré své požadavky ohledně cíle a výstupů projektu na samotném začátku. Dále se očekává, že se s definovanými požadavky už nebude dále nijak významně manipulovat či je ve větší míře měnit. Následujícím předpokladem je, že projektový tým je s danou problematikou obeznámen a přesně ví, jakým způsobem chce požadovaného cíle dosáhnout. To znamená, že projektový tým je schopen předem určit nástroje a metodiku, které bude k úspěšnému zhotovení projektu potřebovat, na základě čehož je možné každé fázi projektu předem přidělit nezbytné zdroje. S ohledem na všechny tyto skutečnosti pak nic nebrání vzniku detailního plánu, který po zbytek projektu slouží jako stěžejní vodítko a ukazatel úrovně rozpracování. Tradiční přístup je tedy někdy považován za plánem řízený přístup, anglicky plan-driven approach [6].

Plán je alfou i omegou celého projektu, což je výhodou i nevýhodou zároveň. Výhodou takového přístupu je předem daná posloupnost jednotlivých procesů a jednoduchá kontrola, zdali úroveň rozpracování odpovídá plánu, či za ním zaostává. Detailní plán navíc vlévá do projektového řízení určitou úroveň jistoty a snižuje riziko výskytu nečekaných událostí. Nevýhodou pak je bezesporu netolerance ke změnám, přičemž faktorem úspěchu je vyhovění plánu, nikoliv dodání přidané hodnoty zákazníkovi a jeho výsledná spokojenost. Shrneme-li tyto poznatky, uplatnění tradičního přístupu se nabízí u projektů podobným těm, které byly již v minulosti úspěšně vypracované a u kterých nehrozí nečekané změny či překvapení. Takové projekty jsou již detailně zmapovány a zkušený projektový tým je schopen vypracovat plán natolik kvalitní, že může sloužit jako stěžejní vodítko celého projektu.

Lze říci, že tradiční přístup má pořád své místo v projektovém řízení, nicméně neochota přijímat změny zdaleka neodpovídá dnešnímu dynamickému a rychle se vyvíjejícímu podnikovému prostředí, ve kterém se organizace a jejich projektové týmy nachází. Robert Wysocki [6, s. 42] ve své knize uvádí na základě svého průzkumu, že dnes je řízeno okolo pouhých 20 % všech projektů tradičním způsobem a jedná se z velké části právě o ty projekty, se kterými už existuje bohatá zkušenost a jejichž implementace není nikterak vzdálená od projektů, jež byly v minulosti již úspěšně implementovány. Wysocki očekává, že podíl tradičního přístupu v projektovém řízení bude i nadále upadat na úkor modernějších metodik, které se řadí do tzv. agilního přístupu, jenž se více zaměřuje na zákazníka a upřednostňuje jeho přidanou hodnotu před striktním dodržováním plánu. Na druhé straně projektový manažer s více než 25letou praxí Thomas Franchina [8] s názorem Wysockého, že se tradiční přístup dnes objevuje už pouze zřídka, nesouhlasí. Franchina argumentuje tím, že právě opakovanost, která je typická u tohoto přístupu, je důvodem dnešního intenzivního využití TPM například u developerských projektů, maloobchodních řetězců nebo fastfoodů, kde je realizováno velké množství prakticky totožných projektů na základě již existujících prototypů, které slouží jako kompletní předloha. Na tomto příkladu je dobře vidět, že využití konkrétních metodik a přístupů velmi záleží na charakteru projektu a odvětví, ve kterém je daný projekt realizován.

3 Tradiční metodiky projektového řízení

Tradiční metodiky a metody projektového řízení se zabývají konkrétní aplikací principů tradičního přístupu v praxi. Tyto metodiky obvykle vychází z pěti základních fází životního cyklu projektu, jimiž jsou [3]:

- iniciace,
- plánování a návrh,
- realizace,
- monitorování a kontrola,
- uzavření.

Tradiční metodiky jsou postaveny na předpokladu, že skutečnosti, které by mohly ovlivnit vývoj projektu, jsou předvídatelné, a projektový tým skvěle rozumí všem procesům a činnostem, které jsou za účelem vyhotovení projektů vykonávány [9]. Většinou mohou být tyto předpoklady splněny spíše u méně komplexních projektů, na kterých pracují projektové týmy se zkušenostmi s podobnými zakázkami z dřívějška [6, s. 42-45], nicméně není to podmínkou.

Přejdeme-li již ke konkrétním metodikám. V této práci se budeme věnovat především typickým představitelům tradičního přístupu, jimiž jsou Vodopádový model a Spirálový model, které stály na úplném počátku vývoje projektových metodik, kde představovaly převrat v oblasti projektového řízení, především pak v projektech zabývajících se vývojem softwaru [10, s. 55]. Na závěr této kapitoly si přiblížíme také metodiku Rational Unified Process (RUP) od společnosti IBM jako příklad komplexní komerční metodiky, která je také postavena na tradičních principech.

3.1 Vodopádový model

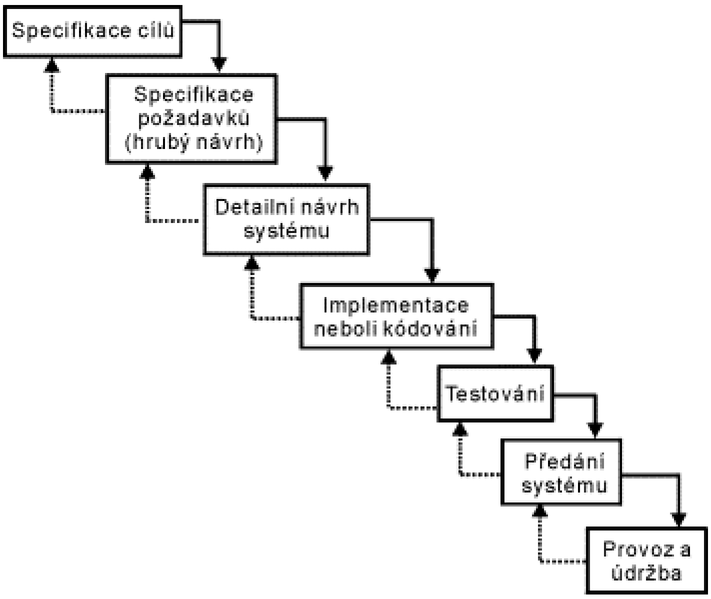
Na začátku 70. let minulého století Dr. Winston Royce definoval původní Vodopádový model[1], který přímo vycházel z jeho předchůdce, tzv. Stagewise modelu. Oba tyto modely byly původně navrženy pro projekty zabývající se vývojem softwaru, nicméně jejich principy lze uplatnit i v ostatních oblastech projektového řízení [6, s. 368]. Model

[1] Dnes existuje mnoho variant a modifikací Vodopádového modelu, které řeší zásadní nedostatky modelu původního. Příkladem je Rapid Development Waterfall model nebo Modified Waterfall model.

Stagewise byl založen na striktní posloupnosti fází a postrádal jakoukoliv formu zpětné vazby. Právě absenci zpětné vazby pak vyřešil příchod Vodopádového modelu.

Ve skutečnosti se ani u jednoho z těchto modelů nejedná o metodiku, jako spíše o model životního cyklu projektového řízení. Avšak chápeme-li metodiku jako jakýsi návod či rámec pro řízení projektů za účelem zvýšení efektivity a jeho úspěšného dokončení, pak se dá v určitém slova smyslu považovat i model životního cyklu projektového řízení za metodiku [10, s. 55].

Vodopádový model přistupuje k projektu jako k posloupnosti po sobě jdoucích etap, které jsou zpracovány sekvenčně jedna za druhou od shora dolů, viz Obrázek 2. Jedná se tedy o sekvenční model, který je vhodný např. pro konstrukční či montážní projekty [6, s. 368]. Podmínkou pro započetí prací na nové etapě je kompletní dokončení předchozí etapy, jejíž výstupy poslouží zároveň jako vstupy pro procesy z etapy po ní následující. Jakmile je jednou nějaká fáze projektu ukončena, nemělo by se do ní už nijak významně zasahovat. Model sice nabízí limitovaný prostor pro úpravy mezi etapami, ale k těm by mělo docházet pouze výjimečně [11, s. 82-85].



Obrázek 2: Vodopádový model
Zdroj: Životní cyklus informačního systému [online]. [cit. 2016-11-20]. Dostupné z: http://www.fi.muni.cz/~smid/image302.gif

Součástí modelu je i validace a verifikace jednotlivých etap, čímž se ověří správnost produktu a jeho vývoje [12, s. 79]. Existence této zpětné vazby, jež je na obrázku znázorněna tečkovanými šipkami, je hlavní rozdíl oproti staršímu Stagewise modelu, kde chybí.

3.1.1 Hodnocení metodiky Vodopádový model

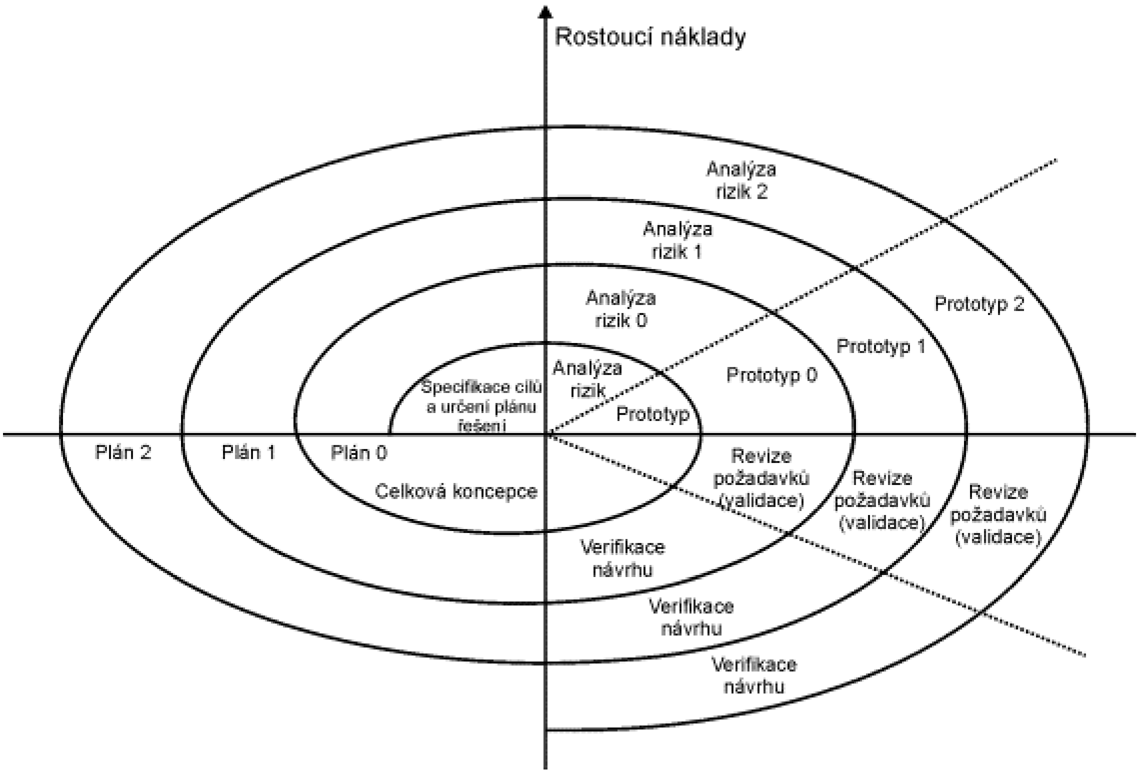
Sekvenčnost modelu umožňuje projektovým manažerům velmi detailně naplánovat jednotlivé etapy a sledovat jejich včasné plnění, či případné zpoždění oproti plánu. Kvalitní naplánování a specifikace požadavků také umožňují relativně přesně stanovit výši nákladů projektu v jeho počátcích [13]. Vodopádový model je vhodný pro srozumitelné projekty s jasně daným cílem a technologickým řešením, kde se neočekávají pozdější změny v požadavcích zákazníka. Naopak u projektů, kde je dynamické prostředí a vysoké riziko pozdějších změn, je tento model krajně nevhodný a měla by být zvolena jiná, vhodnější a flexibilnější metodika. Významnou slabinou Vodopádového modelu je minimální komunikace projektového týmu se zákazníkem, k té dochází nejvíce na začátku projektu při specifikaci požadavků a pak na samotném konci při předávání finálního produktu. Zákazník během vývoje zpravidla nemá přístup k žádným prototypům a v případě nepochopení některých stran může dojít k vývoji produktu, jenž neodpovídá zákazníkovým představám.

3.2 Spirálový model

Spirálový model, obdobně jako Vodopádový model, je spíše životním cyklem projektového řízení nežli metodikou, nicméně lze aplikovat stejnou myšlenku jako v předchozím případě a v určitém slova smyslu jej za metodiku považovat. Stejně jako jeho předchůdce i Spirálový model vznikl v oblasti softwarového vývoje, kde je považován za revoluční mezník, co se přístupu k vývoji týče [10, s. 66]. Samotný model byl představen v polovině 80. let jako nástupce Vodopádového modelu, jenž se tou dobou začal jevit jako nepříliš vhodný pro komplexnější a složitější projekty. Spirálový model pak nově přinesl do oblasti vývoje softwaru revoluci v podobě iterativního přístupu, který se promítá do absolutní většiny dnešních metodik a bez kterého si lze jen těžko představit v současnosti vývoj jakéhokoliv komplexního produktu [10, s. 67].

Na rozdíl od Vodopádového modelu, který je postaven na dokonalém porozumění požadavků ze strany zákazníka, se Spirálový model hodí pro projekty nebo jejich části, kde specifikace nejsou na začátku zcela jasné a jejich formulace je složitá. Příkladem takového projektu může být vývoj nové technologie. V takovém případě je výhodou evolučnost tohoto modelu, kdy jsou jednotlivé procesy opakovány a s každou další iterací podávají přesnější a kompletnější výsledky blížící se finálním požadavkům klienta. Tímto způsobem klient vidí postupný vývoj svého produktu, a může s ohledem na jeho aktuální stav upravovat své požadavky či přidávat požadavky nové do budoucích iterací [14, s. 96]. Spirálový model do oblasti projektového řízení dále přinesl analýzu rizik, které zároveň přikládá velkou důležitost, a proto se také tento model někdy řadí mezi tzv. riziky řízené přístupy, anglicky risk-driven approaches [10, s. 67].

Samotný projekt v rámci modelu začíná ve středu spirály a vyvíjí se směrem ven, viz Obrázek 3. Čím více iterací projekt má, tím delší bude linka spirály a tím delší a nákladnější bude ve výsledku i celý projekt.



Obrázek 3: Spirálový model
Zdroj: Životní cyklus informačního systému [online]. [cit. 2016-11-20]. Dostupné z: http://www.fi.muni.cz/~smid/image306.gif

3.2.1 Hodnocení metodiky Spirálový model

Spirálový model může dobře fungovat u projektů, které nejsou detailně specifikovány a jejichž zadání stále nemusí být definitivní. Produkt je v tomto případě vyvíjen přirozeně –

evolučně a detailní specifikace na začátku není nezbytná. Nicméně v každé iteraci by měl být jasný cíl, kterého se má v daném cyklu dosáhnout, a tento cíl pak s každou další iterací upravovat směrem k finálnímu produktu [14, s. 96]. Hlavní přínos iterativního přístupu pak spočívá v tom, že zákazníkovi umožní vidět vyvíjené prototypy a reagovat na ně. Na druhé straně s sebou nese tento model i několik rizik, kdy může prototyp vyvolat v zákazníkovi mylnou představu o fázi rozpracování projektu tím, že zákazník nevidí procesy na pozadí. Další hrozbou může být také nadměrný počet iterací, následkem čehož se projekt zbytečně protahuje a dochází k jeho prodražení. Kvalitní řízení a kontrola iterací jsou tudíž nezbytnou součástí celého modelu. Naopak analýza rizik, na kterou je zde kladen velký důraz, umožňuje včasné vyloučení nesprávných řešení, čímž se minimalizují potenciální časové a finanční ztráty plynoucí z rizikových částí projektu [10, s. 73].

Závěrem lze říci, že model lze použít pro malé i rozsáhlé projekty, nicméně u malých projektů může složitost modelu působit těžkopádně a zbytečně komplikovaně. V dnešní době je tento model používán čím dál méně a nahrazují ho sofistikované metodiky, které ze Spirálového modelu vycházejí [10, s. 77], příkladem takové metodiky je Rational Unified Process (RUP) společnosti IBM, které se věnuje následující kapitola.

3.3 Metodika Rational Unified Process

Metodika Rational Unified Process je komerčním produktem společnosti IBM, nicméně u jejího zrodu stála společnost Rational Software Corporation, kterou společnost IBM v roce 2003 převzala. Metodika od jejího počátku podléhala aktivnímu vývoji, který vedl k vydávání novějších verzí, dnes je však vývoj již ukončen. Poslední verzí, kterou společnost IBM certifikovala, pak byla verze RUP 7.0, přičemž její certifikace byla ukončena v polovině roku 2016 [15]. V roce 2006 vytvořila IBM na základech RUP alternativní metodiku s názvem Open Unified Process, jež kombinuje principy RUP a agilního přístupu k vývoji softwaru [16].

RUP je komplexní metodika vyvinutá výhradně pro projekty z oblasti softwarového vývoje. V praxi slouží jako podrobný soubor pravidel a praktik k efektivnímu vývoji softwaru, přičemž tento ucelený rámec je distribuován mezi všechny účastníky daného projektu formou internetové znalostní báze. Tato znalostní báze bohatě využívá možností jazyka UML pro popsání a vykreslení obsažených procesů a dále poskytuje dílčí nástroje, dokumentace a šablony [10].

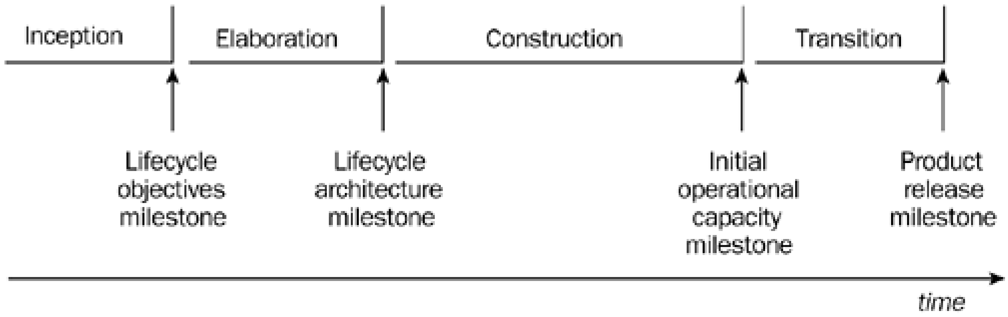
Základním elementem celé metodiky je tzv. případ použití, anglicky use-case, který lze definovat jako posloupnost kroků v interakci mezi uživatelem a systémem prováděné za určitým účelem a s přidanou hodnotou uživateli [17]. Stěžejní principy pak shrnuje šest klíčových praktik, které by měl projektový tým následovat a které přispívají k efektivnímu vývoji systému. Dané praktiky jsou podle oficiálního dokumentu společnosti Rational – RUP Best Practices [18] následující:

- **Iterativní vývoj softwaru** – současná softwarová řešení jsou většinou tak komplikovaná, že je téměř nemožné je úspěšně a kompletně implementovat na první pokus. Jako vhodnější řešení se tedy nabízí vyvíjet tyto složité systémy v etapách – iteracích. Postupný vývoj v rámci těchto iterací je zakončený vždy funkčním prototypem, který se s přibývajícími iteracemi čím dál více přibližuje k finálnímu produktu. Tento způsob umožňuje zákazníkovi vyzkoušet aktuální verzi softwaru na konci každého cyklu a případně vydat změnový požadavek k jeho úpravě za předpokladu, že neodpovídá jeho představám. V neposlední řadě dochází k frekventovanému testování, jelikož během každé iterace se musí nově vzniklý prototyp kompletně otestovat.
- **Správa požadavků** – metodika vyzdvihuje důležitost správy požadavků zákazníka, které udávají finální podobu celého softwaru. V případě, že dojde ke změně, je nezbytné změnový požadavek zapracovat tak, aby se celý vývoj i následné testování řídilo podle aktuálně chtěné zákazníkovy verze. Takto je dosaženo přehledného systému, kde jsou dostupné informace ohledně zadání projektu a jeho případných změnách. Na základě těchto informací probíhá vývoj a testování celého projektu.
- **Použití komponentové architektury** – RUP podporuje využití komponentové architektury, která umožňuje efektivní využití již existujících samostatných softwarových řešení, tzv. komponent, v rámci nově vznikajícího systému. Tyto komponenty jsou poskládány na základě zvolené architektury. Komponentovou architekturou lze dosáhnout významné úspory zdrojů a jednodušší modifikace,

jelikož případná změna se může týkat pouze dané komponenty a nikoliv zasahovat do celého systému.

- **Vizuální modelování softwaru** – správná volba architektury, komponent a výsledná správná implementace ve velké míře závisejí na správném pochopení a porozumění celého systému a jeho vývoje. Za tímto účelem se vytváří grafická abstrakce s pomocí modelovacího jazyka Unified Modeling Language (UML), který je modelovacím standardem metodiky RUP. Za účelem modelování jsou definovány čtyři základní prvky, jež zároveň odpovídají na čtyři základní otázky popisu procesů „Kdo udělá Co, Jak a Kdy?“:
    - Kdo? – definice rolí a pracovníků,
    - Jak? – definice činností,
    - Co? – definice artefaktů, které reprezentují formální produkty a dokumenty,
    - Kdy? – definice pracovních procesů.
- **Ověřování kvality softwaru** – kvalita s ohledem na výkonnost a spolehlivost systému je velmi důležitá a její kontrola je nezbytnou součástí samotného procesu vývoje softwaru. RUP integruje ověřování kvality přímo do procesu vývoje, jinými slovy, kvalita je kontrolována ve všech aktivitách a všemi účastníky.
- **Řízení změn** – změny během projektu jsou prakticky nevyhnutelné a jejich řízení je důležitou součástí úspěšné implementace. Metodika popisuje, jak změny řídit napříč projektem, čímž je dosaženo aktuálnosti dokumentace, modelů i samotného kódu.

RUP využívá iterativního přístupu, přičemž každá iterace je rozdělena do 4 fází a každá fáze je ukončena předem definovaným milníkem [19], viz Obrázek 4. Milníky slouží ke stanovení cíle každé fáze. Určují okamžik v čase, kdy se vyhodnocují konkrétní výsledky, dosažení stanovených cílů a rozhoduje se o budoucím vývoji projektu [20]. Životní cyklus vývojové etapy systému začíná zahájením (Inception), během něhož se stanoví cíle, analyzují rizika a specifikují hlavní případy užití, role a ostatní prvky systému. Dále se vypracuje časový harmonogram a očekávané náklady. Následující fází je rozpracování (Elaboration), v němž se specifikuje architektura, a jsou vytvořeny první prototypy. Fáze rozpracování je kritická z hlediska rozhodnutí, zdali projekt ukončit, či v něm pokračovat. Během realizace (Construction) je celý systém navržen, naprogramován a paralelně testován. Konečnou fázi je nasazení (Transition), kdy je produkt předán do další iterace nebo nasazen k ostrému provozu. V takovém případě může být součástí i školení uživatelů a vytvoření uživatelské dokumentace [19, s. 37].



Obrázek 4: RUP - Dělení iterace na 4 základní fáze s milníkem na konci každé fáze
Zdroj: RUP Milestones [online]. [cit. 2016-11-20]. Dostupné z: http://flylib.com/books/en/4.311.1.100/1/

3.3.1 Hodnocení metodiky RUP

RUP je velmi komplexní metodika, která klade důraz na detailní dokumentaci, z čehož plynou její silné stránky, stejně jako její nevýhody a těžkopádnost. Využitím komponentové architektury lze ušetřit nemalé zdroje při vývoji. Za zmínku stojí také znalostní báze, která je distribuována přes internet a která umožňuje jednoduchý přístup projektového týmu k metodickým předpisům. Nicméně osvojení metodiky vyžaduje nemalé množství času a finančních prostředků, proto je vhodná pro dlouhodobé a stálé vývojové týmy, kterým lze vyčlenit dostatek času a prostředků na zavedení metodiky [10, s. 96]. Naopak pro týmy, jež vznikly za účelem jednorázového projektu, je tato metodika vzhledem k její složitosti nevhodná. Metodika se dá přizpůsobovat a konfigurovat, aby vyhovovala konkrétním řešením, avšak je vhodnější spíše pro větší projekty a větší vývojové týmy, u kterých je vyžadována vysoká úroveň standardizace a podrobná dokumentace. Ovšem metodika je natolik adaptabilní, že ji lze využít i v případě menších týmů a menších projektů.

4 Agilní přístup k projektovému řízení

V předchozí kapitole jsme si představili tradiční přístup spolu se třemi konkrétními metodikami. Máme-li hrubou představu o tradičním přístupu a jeho metodikách, je ta pravá chvíle začít se podrobněji věnovat agilnímu směru. V této kapitole se celou problematikou agilního přístupu budeme zabývat v obecné rovině a přes Manifest agilního vývoje softwaru se seznámíme s klíčovými principy agilního směru. V následující kapitole se pak podíváme podrobněji na aplikaci těchto principů v praxi skrze konkrétní metodiky.

Agilní přístup, tak jak jej popsal Wysocki [6, s. 48] ve své knize, leží někde mezi tradičním a extrémním přístupem, tedy v situaci, kdy máme dobře definovaný cíl, avšak cesta k jeho dosažení není zcela jasná. V takovém případě není možné vytvořit detailní plán, který je klíčový pro tradiční přístup a vzhledem k jasně definovanému cíli se nejedná ani o jeden z extrémních přístupů.

Agilní projekty sdílejí několik charakteristických rysů. Jak již bylo řečeno, jedná se o projekty, jež mají stanovený cíl, ale řešení, jak jej dosáhnout, je nejasné. Neznáme-li postup, pak nelze práce na projektu nijak naplánovat, a nezbývá než se do projektu pustit takříkajíc „po hlavě“. Jinými slovy jedinou cestou vedoucí k cíli je přístup, jenž umožní najití daného řešení za běhu [6, s. 48]. V takovém případě je důležitá úzká spolupráce s klienty proto, aby dávali frekventovanou zpětnou vazbu k odvedené práci a na jejím základě správně nasměrovali budoucí vývoj projektu či jej případně včas ukončili. Klienti se tak prakticky stávají součástí projektového týmu a přímo se podílejí na vývoji. Na rozdíl od TPM, jenž je řízen plánem a změny považuje za krajně nežádoucí, agilní přístup změny vítá a považuje je za nezbytnou součást projektového řízení vedoucího k dosažení maximální přidané hodnoty pro zákazníka. Také proto je agilní přístup někdy označován jako přístup řízený změnou, anglicky change-driven approach [6].

Absence detailního plánu, nejisté řešení a řízení změnami se pak mohou zdát jako rizikové faktory, což je ve výsledku pravda. Agilní projekty mohou být pro podniky velmi rizikové, ale často jsou to inovativní a klíčové projekty, které tvoří budoucí obchodní příležitosti či konkurenční výhodu [8]. Jelikož tyto projekty bývají často komplexní, je vhodné za účelem minimalizace rizika rozdělit je na menší dílčí projekty, které budou mít na starosti menší několikačlenné týmy. Typický agilní projektový tým je pak obvykle tvořen nemnoha členy a věnuje se menším projektům, nebo dílčím problémům v rámci komplexního projektu. Takto se minimalizuje riziko a maximalizuje výsledná přidaná hodnota pro zákazníka, která je klíčová [7].

Agilní smýšlení existuje už dlouho, avšak formálně byly agilní principy formulovány až na počátku tohoto tisíciletí formou Manifestu agilního vývoje softwaru.

4.1 Manifest agilního vývoje softwaru

V roce 2001 se v Utahu v USA sešla skupina odborníků z oblasti softwarového vývoje, kteří pracovali na alternativách k tradičním metodikám projektového řízení, jež se zdály být zastaralé a neschopné vyhovět nárokům na rychlost vývoje v 21. století. Výsledkem tohoto setkání byl dokument s názvem Manifest agilního vývoje softwaru, ve kterém byly specifikovány principy agilního, v té době ještě relativně nového, směru. Doslovné znění manifestu v české lokalizaci na jeho oficiálních stránkách [21] je následující:

„Objevujeme lepší způsoby vývoje softwaru tím, že jej tvoříme a pomáháme při jeho tvorbě ostatním. Při této práci jsme dospěli k těmto hodnotám:

- *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*

Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více.“

Manifest nejprve vyzdvihuje interakci jednotlivců před pevně stanovenými procesy a nástroji, které mohou být důležité a užitečné, avšak nesmí být na úkor týmové práce, která bohatě přispívá k budoucímu úspěchu projektu. Tvůrčí týmy si nejlépe samy vyberou, které z nástrojů uplatní a které jim pomáhají k efektivnímu provedení práce. Druhý bod manifestu se zaměřuje na důležitost funkčnosti softwaru a tomu odpovídající dokumentaci. V praxi se stává, že zákazník dostane velmi obsáhlou dokumentaci, avšak samotný software obsahuje nedostatky a ještě není zcela funkční. Cílem dokumentace by mělo být objasnění softwaru a vysvětlení těch oblastí, které nejsou intuitivní a přehledné, a těch by mělo být pokud možno co nejméně. Kvalitní dokumentace je důležitá, avšak agilní metodiky říkají, že by nemělo docházet k tvorbě dokumentace na úkor kvality samotného produktu. Třetí bod je obdobou bodu prvního, avšak na rozdíl od něj se zaměřuje na zákazníka a nikoliv na vývojový tým. Opět vyzdvihuje nutnost spolupráce, která je nezbytným prostředkem pro kvalitní výstup projektu a spokojenost obou stran. Jakýsi rámec této spolupráce určuje smlouva, jež je bezesporu zásadním dokumentem, ale je nutné se už při jejím podpisu zamyslet nad případnými změnami a dopady těchto změn na projekt samotný. Smlouva by pak spíš měla udávat mantinely celého projektu, nikoliv definovat jeho průběh a výstup do posledního detailu. V konečném důsledku nebrat ohledy na změnové požadavky zákazníka a odkazovat se na smlouvu by mohlo mít za následek ztrátu hodnoty konečného produktu, nespokojenost zákazníka a rozvázání potenciální budoucí spolupráce [22]. Poslední bod manifestu je obdobou toho předchozího, pouze se týká plánu a nikoliv smlouvy. V zásadě vyzdvihuje, jak důležité je u projektu vyjít vstříc měnícím se trendům a požadavkům zákazníka i přesto, že se to nemusí shodovat s plánem. Plán slouží jako určité vodítko projektu, avšak v dnešním dynamickém světě může 100% dodržování plánu na úkor požadavků zákazníka vést ke zkostnatělosti a neúspěchu projektu. Jinými slovy změnám v dnešní době nelze tak docela zabránit a s největší pravděpodobností nastanou, proto je lepší a efektivnější se na ně připravit a vyjít jim vstříc.

Kromě základní formulace manifest obsahuje ještě 12 základních principů, které vystihují základní myšlenky agilního vývoje softwaru a agilního přístupu k projektovému řízení obecně, viz Příloha A. S ohledem právě na tyto principy lze agilní projektové řízení popsat následovně.

- Na prvním místě je fungující řešení, které je zákazníkům dodáno včas a které jim přináší užitnou hodnotu. Tato užitná hodnota vychází právě z funkčnosti celého řešení, nikoliv z bohaté dokumentace a množství vývojových diagramů.
- Změny ze strany zákazníka jsou akceptovány, a to i v pozdějších fázích vývoje, jelikož mohou vést v konečném důsledku k jeho konkurenční výhodě.
- Vývoj probíhá v krátkých iteracích od 2 týdnů až do 2 měsíců, kdy na konci každého cyklu je samostatně fungující inkrement, jenž zákazníkovi přináší přidanou hodnotu. Na rozdíl od tradičních přístupů je u agilního přístupu tendence tyto cykly zkrátit na minimum.

- Vývojový tým a zákazník jsou v blízkém kontaktu a komunikují na denní bázi. Zpočátku jsou definovány pouze hrubé požadavky, které jsou následně upřesňovány a pozměňovány na základě probíhající komunikace.
- Klíčovým faktorem úspěchu projektu je motivovaný tým a jedinci v něm. Je nezbytné udržet zdravé pracovní prostředí, kdy zúčastnění pociťují osobní zájem na úspěchu projektu a tento zájem shora podporovat.
- Komunikace v rámci týmu a se zákazníkem hraje zásadní roli, přičemž osobní konverzace je nejúčinnější. Agilní metodiky neprodukují stohy dokumentace, jelikož účelem dokumentace je porozumění problému, čehož je dosaženo mnohem rychleji a snáze přímou osobní komunikací.
- Primárním měřítkem úspěchu je fungující software, resp. řešení, ostatní faktory jsou druhotné.
- Z hlediska vývoje je důležitý trvale udržitelný vývoj. To znamená, že práce vývojářů by neměla přesahovat 40 hodin týdně a přesčasy či práce o víkendech by měly být opravdu jen výjimečnou záležitostí. Taková řešení sice krátkodobě produktivitu zvýší, ale z dlouhodobého hlediska dochází k přepracování pracovníků a snižování produktivity.
- Kvalitnímu návrhu a designu je přisuzována velká důležitost. Vytvoření návrhu není samostatnou etapou, naopak jedná se o činnost prolínající se do všech fází projektu. V agilním přístupu je obvykle návrh s přibývajícími změnami upravován.
- Jednoduchost je na prvním místě. Charakteristická je snaha o maximalizaci práce, kterou není třeba vykonávat. Jinými slovy, jednoduché procesy jsou preferovány před složitými a komplikovanými.
- Samoorganizující se týmy jsou schopné vyprodukovat ty nejlepší návrhy a architektury, avšak je třeba jejich kreativitu podporovat důvěrou shora a častou komunikací.
- Vývojové týmy se neustále snaží o optimalizaci procesů a přemýšlí, jak proces vývoje i projektového řízení zdokonalit.

[10] [19] [21]

Manifest i celý jeho obsah se zaměřuje na problematiku vývoje softwaru. Když se bavíme o agilních přístupech k projektovému řízení v obecnější rovině, je nezbytné na principy zmíněné v manifestu nahlížet z trochu jiné perspektivy. Obecně lze říci, že agilní přístupy projektového řízení si dávají za cíl pomoci projektovým týmům reagovat na nepředvídatelné a měnící se požadavky, stejně jako tomu je v případě agilního vývoje softwaru. Konkrétně je toho pak dosaženo například přímou komunikací se zákazníky namísto nadbytečné dokumentace nebo organizací práce do krátkých iterací s jasně definovaným výsledkem [23]. Vzhledem k odlišnému charakteru každého projektu je jasné, že ne všechny body manifestu lze aplikovat na všechny typy projektů, nicméně agilní řízení je především o přístupu a lze jej uplatnit pro řízení velké škály různých projektů nad rámec softwarového vývoje.

5 Agilní metodiky projektového řízení

Konkrétní aplikací principů agilního přístupu se zabývají sofistikované metodiky, které se dnes ve světě projektového řízení těší čím dál větší oblibě. „Agilní“ v překladu znamená flexibilní, rychle reagující, přizpůsobivý či dynamický. To jsou vlastnosti, které jsou v současnosti nezbytné, má-li být projekt úspěšně implementován, jelikož jak se říká – jedinou jistotou je změna, a v dnešním dynamickém a moderním světě to platí dvojnásob.

Agilní metodiky kladou velký důraz na spolupráci a komunikaci lidí v týmu, stejně jako se zákazníky. Kýženým výsledkem je především funkční produkt, nikoliv stohy dokumentace a analýz. V této kapitole si představíme konkrétní metodiky, které z těchto předpokladů vychází, detailněji se budeme zabývat metodikami Scrum, Lean Development, Extreme Programming a Dynamic Systems Development Method. Většina těchto metodik je zaměřena na projekty týkající se vývoje softwaru, jelikož obdobně jako u tradičních metodik, právě v oblasti softwarového vývoje agilní smýšlení započalo. Agilně se však bez pochyby dají řídit také ostatní projekty, jež se nutně nemusí zabývat softwarovou problematikou [24]. Jak uvádí Jim Highsmith [25], spoluautor agilního manifestu a projektový manažer s více než 30letou praxí:

„Agilita je schopnost vytvářet a reagovat s cílem tvorby zisku v proměnlivém a turbulentním podnikovém prostředí. Je to schopnost vyvážit stabilitu a flexibilitu.“

Highsmith přistupuje k agilnímu přístupu obecněji a v této definici agility záměrně vynechává vývoj softwaru, jelikož agilně lze přistoupit k jakémukoliv projektu, jenž takový přístup vyžaduje. Problematikou aplikace agilních metodik se zabývá kromě jiných i Robie Mac Iver, který ve svém článku popisuje alternativní uplatnění metodiky Scrum nad rámec vývoje softwaru. Metodikou Scrum i její aplikací vně oblast vývoje softwaru se zevrubně zabývá následující text.

5.1 Scrum

První zmínky a počátky této metodiky se datují do poloviny 90. let, kdy Ken Schwaber a Jeff Sutherland poprvé veřejně představili koncept metodiky Scrum, na jejímž rozvoji se později podílel také Mike Beedle. Právě tyto tři osobnosti, kromě jiného jedny z klíčových postav a prvních signatářů Agilního manifestu vývoje softwaru, jsou považovány za autory této metodiky. Název „Scrum“ pak pochází z anglického jazyka, kde tento termín znamená tzv. skrumáž, čímž se v ragby označuje shluk hráčů jednoho týmu, kteří se snaží společnými silami dostat míč na žádoucí pozici. Analogicky se snaží i projektový tým s uplatněním metodiky Scrum dovést společnými silami projekt do zdárného konce.

Jak už to u agilních metodik bývá, i metodika Scrum je zaměřena především na vývoj softwaru, nicméně ve výsledku nic nebrání aplikaci této metodiky pro řešení jakéhokoliv projektu, k jehož úspěšnému dokončení se dá aplikovat tento sofistikovaný iterativní přístup. Na oficiálních internetových stránkách společenství Scrum Alliance, což je největší profesní uskupení a certifikační autorita v oblasti agilního projektového řízení, jsou uvedeny příklady, kdy byl Scrum použit k řízení univerzitních projektů, v automobilovém průmyslu nebo dokonce i pro armádní účely [26].

5.1.1 Základní charakteristika

Metodika Scrum řeší problematiku celého životního cyklu projektu od jeho zahájení až po jeho oficiální ukončení. V rámci vývojové fáze jsou zavedeny krátké iterace, obvykle ne delší než 30 dní, které se nazývají Sprinty. Scrum nicméně vychází z předpokladu, že vývojová část projektu je proces především empirický a je nezbytné k němu tak přistupovat [19, s. 55]. Sprinty tedy vyjma formálních schůzek nemají definované žádné specifické procesy, jelikož metodika vychází z přesvědčení, že není možné přesně určit okolnosti vývoje a tím pádem předem stanovit vývojové procesy a způsob, jakým bude projektový tým pracovat [10, s. 148]. Procesní hledisko vývoje je pak možné vyřešit s ohledem na již integrované a osvědčené podnikové procesy nebo lze Scrum skloubit s další agilní metodikou, která naopak řeší pouze problematiku vývoje produktu a samotným řízením projektu se nezabývá. V případě vývoje softwaru by se mohlo jednat třeba o metodiky Extrémní programování nebo Vývoj řízený testy. Více o Sprintu i formálních schůzkách si řekneme později, teď se podíváme blíže na projektový tým a jednotlivé role, které jsou v rámci metodiky specifikovány.

5.1.2 Role a Projektový tým

Metodika obecně dělí osoby zainteresované do projektu na dvě kategorie, tzv. Pigs a Chickens[2]. Rozlišení těchto dvou skupin probíhá na základě toho, zdali jsou dotyční přímo součástí projektového týmu a participují se na vývoji produktu či projektovém řízení, v takovém případě se jedná o Pigs. Pigs na projektu pracují a jsou přímo zodpovědní za výsledek, řadí se mezi ně Product Owner, Scrum Master a všichni členové projektového týmu [27]. Na druhé straně za Chickens se dají považovat klienti, konzultanti a všichni ostatní, jichž se projekt týká, avšak úspěšná realizace projektu není primární náplní jejich práce, respektive nenesou přímou zodpovědnost za výsledek. Z hlediska Scrumu jsou důležití především Pigs, konkrétně pak manažerské role Product Ownera a Scrum Mastera.

Product Owner

Product Owner tvoří pomyslné pomezí mezi zadavatelem a řešitelem projektu. Tato osoba je zodpovědná za správu Product Backlogu[3], a tím i za finální podobu výstupu projektu. Product Owner funguje jako spojka mezi projektovým týmem a zákazníkem. Jeho náplň práce tvoří zajištění komunikace s klienty a tlumočení požadavků a případných změn projektovému týmu. Musí zajistit, aby všichni z projektového týmu rozuměli daným požadavkům a mohli je tak úspěšně implementovat [28]. Product Owner dále definuje priority jednotlivých položek z Product Backlogu, čímž má zajistit návaznost jednotlivých Sprintů a činností tak, aby optimalizoval práci projektového týmu. Nicméně to, jak budou položky implementovány z technického hlediska, již závisí na samotných pracovnících z projektového týmu. Z procesního hlediska je tedy Product Owner vedoucím projektového týmu, avšak v technických otázkách by měl respektovat a naslouchat projektovému týmu v čele se Scrum Masterem [27].

Scrum Master

Scrum Master je unikátní rolí metodiky Scrum, která má zajistit, aby všichni zainteresovaní lidé v projektu, ale i v organizaci, tzn. Pigs i Chickens, rozuměli principům a pravidlům Scrumu a také se podle nich řídili [28]. Má-li projektový tým za úkol

[2] Označení „Pigs and Chickens“ pochází z komiksového příběhu, kde prasátko a kuře přemýšlí o otevření restaurace s názvem „Ham & Eggs“. Příběh pojednává o rozdílu nasazení a závazku obou zúčastněných na restauraci, z čehož vyplývá i analogie s metodikou Scrum.

[3] Product Backlogu je věnován samostatný odstavec v kapitole Průběh projektu a klíčové artefakty.

implementovat klientovy požadavky, pak Scrum Master má za úkol implementovat samotný Scrum. Tato role dohlíží na správnou aplikaci Scrum principů po celou dobu projektu, především viditelné to pak je během plánovaných schůzek, které Scrum Master všechny moderuje. Mezi další jeho poslání patří podpora a motivace týmu, řešení problémů a ochrana projektového týmu od vnějších vlivů tak, aby se tým mohl soustředit pouze a jen na projekt samotný [27]. Nakonec je vhodné podotknout, že role Scrum Mastera a Product Ownera by neměly být reprezentovány jednou osobou, jelikož by mohlo docházet ke střetu zájmů a neplnění některé z těchto rolí v plném rozsahu [19, s. 34].

Člen projektového týmu - Developer

Kromě Product Ownera a Scrum Mastera je zbytek týmu tvořen pouze „řadovými" členy, tzv. Developers. Tito členové nemají žádné další hodnosti či konkrétní role, všichni jsou si rovni. Jejich náplní práce je především vytváření přidané hodnoty pro zákazníka, přičemž by měli být schopni řídit sami sebe.

Jak jsme si uvedli výše, obě manažerské role mají jasně specifikovaný cíl a účel, který však s řízením projektového týmu nemá moc co do činění. Řízení je tedy ponecháno na projektovém týmu samotném, respektive je to optimální stav, kterého se snaží Scrum Master dosáhnout. Není-li tým schopen řídit sám sebe, pak je na Scrum Masterovi, aby jej částečně řídil, avšak zároveň musí vyvíjet iniciativu, která dovede tým k tomu, aby byl schopen řídit sám sebe [27].

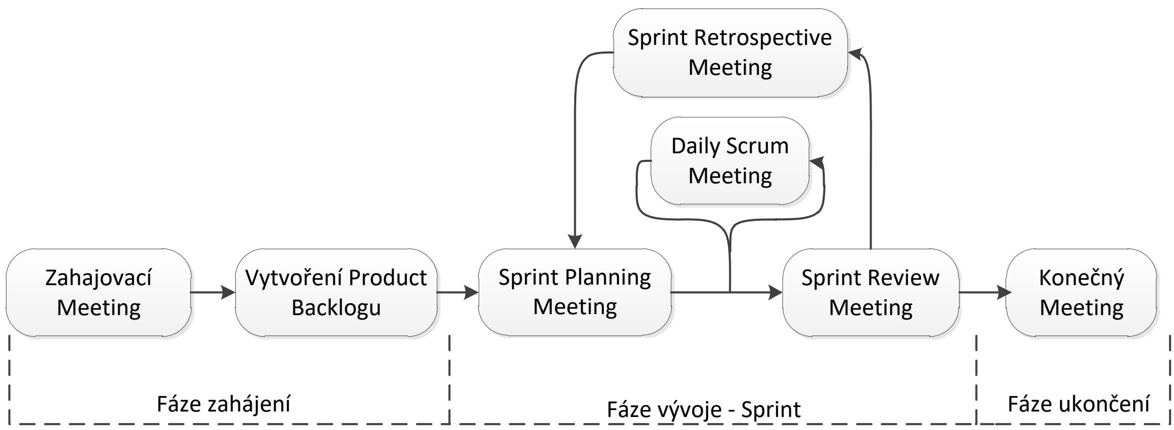
Projektový Tým

Projektový tým by měl být optimálně tak veliký, aby mu nechyběla profesní vybavenost k úspěšnému dokončení složitějších projektů, na druhé straně by měl zůstat dostatečně malý, aby nevznikal chaos a celý projekt se dal empiricky řídit. V praxi se doporučuje vytvářet týmy maximálně do 10 lidí [28]. Tým je z hlediska projektového řízení pouze jeden a řídí sám sebe, jeho členové by měli napříč týmem disponovat všemi nezbytnými dovednostmi, aby byli schopni projekt úspěšně dokončit v celém jeho rozsahu. Uvnitř týmu nedochází k žádnému dělení na sekce či sub-týmy, např. testeři, programátoři atd. Všichni pracují společně na společném cíli, za který jsou také kolektivně zodpovědní [28].

Je-li projekt příliš komplexní, je možné jej implementovat s pomocí několika samostatných Scrum týmů, které pracují paralelně vedle sebe, nicméně působí jako nezávislé jednotky, které řídí samy sebe. V takovém případě musí být kladen zvláštní důraz na to, aby zhotovené projektové inkrementy byly kompatibilní.

5.1.3 Průběh projektu a klíčové artefakty

Jak jsme si už uvedli, celý projekt je členěn na jednotlivé iterace, které jsou ve Scrumu nazývány Sprint. Sprint je základní jednotkou celé metodiky, avšak projekt se neskládá pouze ze Sprintů, ty tvoří výhradně fázi vývoje produktu, někdy označovanou také jako Hra [10, s. 151]. Vývoji předchází fáze zahájení (Předehra) a po posledním Sprintu je celý projekt završen fází ukončení (Dohra), kdy se koná závěrečné setkání s klienty, a finální výstup projektu spolu s veškerou dokumentací a ostatními náležitostmi je představen a předán do rukou zákazníka [27]. Tyto dvě fáze nejsou na rozdíl od Sprintů empirické a sestávají z jasně definovaných procesů [19, s. 55]. Celý životní cyklus projektu dle metodiky Scrum je názorně zobrazen na následujícím obrázku.



Obrázek 5: Životní cyklus projektu a formální schůzky dle metodiky Scrum
Zdroj: vlastní tvorba dle [27]

Během zahájení projektu probíhá seznámení projektového týmu, rozdělení jednotlivých rolí, předběžné plánování a specifikace detailů ohledně produktu, architektury, harmonogramu, zdrojů atd. [27, s. 85]. V této fázi vzniká velmi důležitý artefakt, který bude určovat podobu celého projektu, jedná se o tzv. Product Backlog, který jsme již dříve zmínili ve spojení s Product Ownerem. Z Product Backlogu později vychází další důležitý artefakt – Sprint Backlog. Oba tyto dokumenty si teď blíže představíme.

Product Backlog

Product Backlog je ve skutečnosti seznam či tabulka obsahující veškeré požadavky ohledně výsledného produktu. Dále v něm lze nalézt informace týkající se priority a stavu rozpracování jednotlivých položek. Product Backlog obsahuje zkrátka vše, co se týká finálního produktu a co musí být do konce projektu úspěšně implementováno [28]. Za jeho správu je zodpovědný Product Owner, který musí zajistit aktuálnost, dostupnost a srozumitelnost tohoto artefaktu.

Je zřejmé, že na samém počátku projektu (ve fázi zahájení) nemůže být tento seznam kompletní, jelikož agilní metodiky vycházejí z předpokladu, že není možné veškeré náležitosti specifikovat na počátku, ale k jejich upřesnění dochází během vývoje produktu. Během zahajovací fáze dojde tedy k jakémusi hrubému náčrtu požadavků, které jsou následně během probíhajících Sprintů upřesňovány. Lze říci, že Product Backlog se vyvíjí spolu s produktem samotným [28]. Je-li vytvořený Product Backlog, může se přejít k plánování následujícího Sprintu, tedy k vytvoření Sprint Backlogu.

Sprint Backlog

Spring Backlog je stěžejní dokument z hlediska konkrétních iterací – Sprintů. Definují se v něm cíle daného Sprintu, které jsou vybrané z Product Backlogu. Jedná se tedy o jeho podmnožinu a dají-li se na konci projektu dohromady všechny položky ze všech Sprint Backlogů, měly by se shodovat právě s aktuálním Product Backlogem. Důležité je, že každý Sprint má právě jeden Sprint Backlog, na základě kterého se odvíjí náplň celé iterace. O aktualizaci se stará celý projektový tým. Je-li nějaká práce ze Sprint Backlogu dokončena, musí se její stav aktualizovat a naopak, je-li vyžadována nějaká práce navíc proto, aby bylo možné dosáhnout cíle daného Sprintu, musí být tato položka do Sprint Backlogu přidána [28]. S plánováním nového Sprintu je nezbytně spjato i vytvoření nového Sprint Backlogu.

5.1.4 Sprint a formální meetingy

Sprint je označením vývojové iterace v rámci Scrumu. Jak již bylo řečeno, Sprinty jsou empirické, a to v tom smyslu, že nemají žádné definované procesy a jsou de facto řízeny samotným projektovým týmem na základě toho, co je aktuálně potřeba udělat. Cílem každého Sprintu, obdobně jako u všech ostatních iterací, je vytvoření samostatného produktového přírůstku, jenž přináší klientovi přidanou hodnotu [28]. Délka Sprintu se může lišit od dvou do čtyř týdnů, kratší ani delší iterace se nedoporučují. Je-li v zájmu vývojového týmu předkládat menší části odvedené práce k častější revizi, nebo jsou-li požadavky nestabilní a dochází k jejich častým změnám, pak zvolí tým nejspíše kratší Sprint, zpravidla by však iterace neměly být kratší než dva týdny. V takovém případě by pak mohly formální schůzky významně časově omezovat samostatný vývoj produktu. Naopak je-li v zájmu projektového týmu odevzdávat větší objem práce a jsou-li požadavky relativně stabilní, pak je vhodné zvolit iteraci delší, např. v rozsahu jednoho měsíce, maximálně pak šesti týdnů [27]. Delší iterace by už vedly k celkové těžkopádnosti, zvyšování rizika, pozdnímu odhalování chyb a dlouhým pauzám mezi prezentacemi výsledků jednotlivých Sprintů.

Každý Sprint, ačkoliv je to empirický proces, vychází z Product Backlogu a zachovává určitou základní formální strukturu, viz Obrázek 5. Na počátku je Plánovací Meeting (Sprint Planning), na kterém je vytvořený Sprint Backlog, následně se každý den probíhajícího Sprintu uskutečňují denní schůzky (Scrum Meeting/Daily Scrum). Celý Sprint je z hlediska zákazníka zakončen předvedením výsledné práce v rámci tzv. Sprint Review Meeting. Z hlediska projektového týmu je pak před oficiálním ukončením iterace uskutečněna ještě jedna schůzka s názvem Sprint Retrospective Meeting. Cílem tohoto setkání je optimalizace procesů tak, jak je uvedeno v posledním z 12 základních principů Agilního manifestu vývoje softwaru. V následujícím textu si zmíněné schůzky rozebereme podrobněji.

Daily Scrum / Scrum Meeting

Denní Scrum Meeting je základní schůzkou a charakteristickým prvkem Scrumu. Jedná se o 10 až 15 minut dlouhé schůzky celého projektového týmu, které jsou konány každý den ve stejný čas a na stejném místě. Celou schůzi moderuje Scrum Master, jenž dohlíží na to, aby se vše v daném intervalu stihlo a schůze byla maximálně efektivní. Meeting začne hodnocením Scrum Mastera, který stručně zhodnotí současný stav Sprintu. Za tímto účelem lze použít různé nástroje, často používaný je pak např. Burndown diagram[4]. Nicméně podstatou schůze je především zvýšit celkovou informovanost v týmu ohledně

[4] Burndown diagram je grafické znázornění práce, kterou zbývá udělat oproti času, který zbývá do konce naplánované iterace. Jedná se tedy o přehledný nástroj vypovídající o tom, zdali je projektový tým s pracemi pozadu, napřed nebo zdali je vývoj odpovídající časovým možnostem.

aktuálně odváděné práce a řešit případné problémy. Každý zúčastněný člen podílející se na vývoji produktu tedy musí odpovědět na následující tři otázky:

- Jak a na čem si pracoval od včerejší denní schůze a jak to pomůže týmu k úspěšnému dosažení cíle Sprintu?
- Jak a na čem budeš pracovat do zítřejší schůze a jak to pomůže týmu k dosažení cíle Sprintu?
- Setkal ses s nějakými komplikacemi, které by mohly tobě nebo týmu v budoucnu bránit v dalším postupu směrem k dosažení cíle Sprintu?

Zodpovězením těchto otázek se zvýší povědomí týmu o aktivitách ostatních členů, což pomáhá zvýšit schopnost týmu řídit sám sebe. Tým se tím částečně už sám řídí, jelikož každý plánuje a prezentuje svou práci na následující den [23, s. 18-22]. Na závěr, objeví-li se nějaká komplikace, je v rámci Daily Scrumu na ní upozorněno a společnými silami se tým snaží najít včasné řešení.

Sprint Review Meeting

Sprint Review Meeting je z hlediska zákazníka formálním ukončením aktuálního Sprintu. V rámci tohoto setkání se v neformálním duchu sejde projektový tým spolu se zainteresovanými osobami (Chickens) za účelem prezentace dosažených výsledků Sprintu a aktualizace Product Backlogu [28]. Klienti jsou Product Ownerem obeznámeni s položkami z Product Backlogu, které byly implementovány a které je stále potřeba v budoucích iteracích vypracovat. Od setkání se očekává uvolněná, neformální atmosféra a vzájemná spolupráce všech zúčastněných [23, s. 18-22]. Chickens si mohou prohlédnout a vyzkoušet dosažený produktový přírůstek, během čehož projektový tým očekává zpětnou vazbu ve formě komentářů a připomínek. Připomínky a návrhy zároveň slouží jako podněty k aktualizaci Product Backlogu a plánování dalšího Sprintu [27].

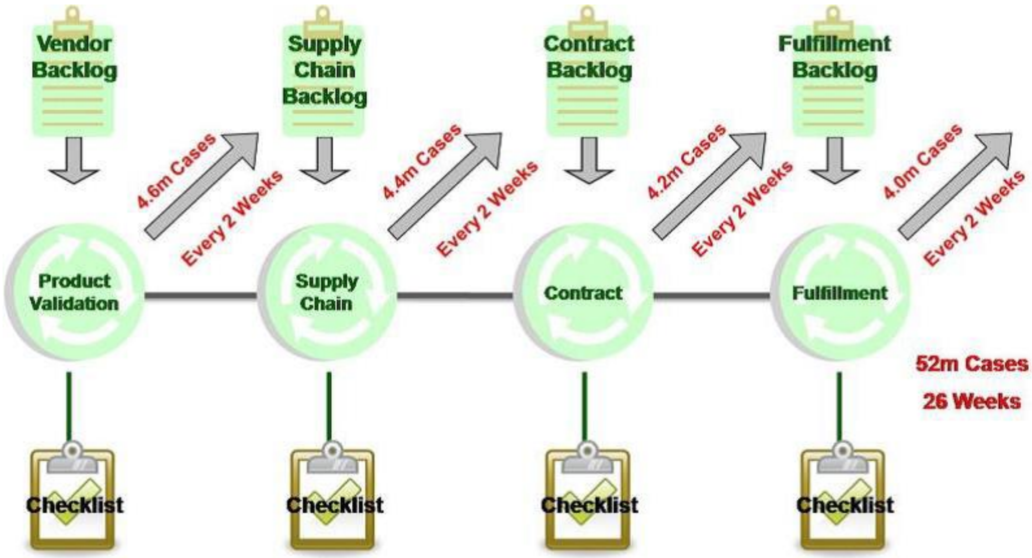
Sprint Retrospective Meeting

Sprint Retrospective Meeting je posledním setkáním v rámci aktuálního Sprintu. Během tohoto meetingu se sejde kompletní projektový tým a snaží se nahlížet zpětně na celý Sprint kritickým pohledem. Cílem je najít slabá místa, především pak procesy, které potřebují změnu a optimalizaci, avšak meeting se zabývá také problematikou samotných lidí, vztahů mezi nimi nebo používanými nástroji [28]. Výsledkem tohoto setkání by měly být odpovědi na otázky *„Co změnit, co zrušit, co ponechat a co nově zavést?"* [27]. Jakmile jsou identifikovaná slabá místa, očekává se od týmu, aby přišel s návrhem na jejich zlepšení a v příštím Sprintu tato řešení integroval.

5.1.5 Příklad aplikace Scrum mimo oblast vývoje softwaru

Jak již bylo řečeno, Scrum byl původně vyvinut pro projekty týkající se vývoje softwaru, avšak lze jej aplikovat i na jiné typy projektů. Takovým případem se zabývá i článek od Robbie Mac Ivera, který stručně představuje aplikaci Scrumu na projekt týkající se naplánování logistického řetězce nového velkoskladu, což zahrnuje zásobování samotného velkoskladu a následnou distribuci produktů stovkám zákazníků. Jedná se o velmi komplexní problém, vzhledem k množství různých produktů i zákazníků a především vzhledem k faktu, že každý ze zákazníků vyžaduje individuální přístup, jelikož má zavedené své vlastní procesy a vyžaduje různé portfolio produktů. Přistoupit k tomuto problému tradičně by bylo nesmírně komplikované a nákladné, jelikož detailně naplánovat projekt na samém počátku pro tak velké množství variabilních požadavků a zákazníků hraničí s nemožným [24].

Na druhé straně Scrum umožňuje sofistikované řešení. Definujeme-li si cílem projektu konkrétní objem produkce proudící velkoskladem za vybrané období, pak lze tento cíl dekomponovat na menší objemy a k těm přistupovat jako k samostatným projektům. Dekompozice komplexního problému na menší celky je často používaný způsob, jak lze agilní přístup na komplexní projekty aplikovat. Mac Iver v jeho případové studii navrhuje dvoutýdenní Sprinty, na kterých se podílí několik jednotlivých Scrum týmů. Konkrétně v tomto případě se jedná o 4 týmy, jež se věnují návrhu a specifikaci produktového portfolia pro jednotlivé zákazníky, návrhu dodavatelského řetězce, smluvnímu ujednání se zákazníky a závěrečnému dokončení, což zahrnuje naskladnění počátečního inventáře a jeho následné doplňování. Za povšimnutí stojí zvolení extrémně krátké iterace, během které by nebylo možné stihnout vše zrealizovat. Tento problém však kompenzuje větší množství týmů, z nichž se každý zaměřuje na určitou oblast původního komplexního problému. Výsledná doba kompletní iterace je pak 8 týdnů, které získáme vynásobením dvoutýdenních Sprintů počtem týmů. Výstup Sprintu jednoho týmu slouží jako vstup pro Sprint týmu druhého a tímto způsobem je práce na určitém množství expedovaného objemu ukončena, jakmile projde všemi Scrum týmy, viz Obrázek 6. Každý tým si současně definuje seznam úkolů, které musí dokončit předtím, než jejich výstup může posloužit jako vstup pro Sprint následujícího týmu.



Obrázek 6: Schéma navrhovaného řešení dle Mac Ivera
Zdroj: **[24]**

Na obrázku vidíme, že Mac Iver si definoval projektový cíl na vytvoření dodavatelského řetězce, kterým projde objem 52 milionů přepravek za rok a který by měl být vytvořen do 26 týdnů od zahájení projektu. Cíl každého Sprintu pak lehce vypočítáme vydělením finálního objemu expedice dobou vyhrazenou na projekt, což jsou 2 miliony přepravek týdně, resp. 4 miliony přepravek na dvoutýdenní Sprint. Jelikož dochází k vyřazení některých odběratelů během procesu, je u předchozích Sprintů navýšen cíl Sprintu, aby ve finále bylo zajištěno předsevzaté množství 52 milionů.

Určitě si povšimneme určité analogie s Vodopádovým modelem v navrhované posloupnosti jednotlivých Scrum týmů, kdy výstup Sprintu jednoho týmu slouží jako vstup pro iteraci týmu následujícího. V tomto případě však probíhají práce paralelně a týmy mezi sebou úzce spolupracují, nejedná se tedy o rigidní posloupnost jako u Vodopádového modelu. Obdobně ani specializace týmů neodpovídá úplně principům, o kterých pojednává Scrum, který vychází z předpokladu, že projektový tým má disponovat všemi funkcemi tak, aby byl schopen zdárně dovést projekt k úspěšnému konci [24]. Nicméně chceme-li aplikovat metodiku na různé typy projektů, může být přínosné upravit si ji lehce k obrazu svému. V tomto případě byl rozsah prací k naplánování a implementaci kompletního dodavatelského řetězce pro jednoho obchodníka tak komplexní a různorodý, že Scrum tým o 10 lidech by zbytečně zápolil s komplikovaností problému, čímž by se zvyšovalo riziko selhání Sprintů i celého projektu. Členěním na specializované týmy se ulehčila specifikace cílů jednotlivých Sprintů a významně snížila komplexita práce daných týmů.

5.1.6 Hodnocení metodiky Scrum

Scrum je v dnešní době jedna z nejpoužívanějších agilních metodik [22, s. 28]. Její výhody spočívají ve flexibilitě, která plyne z toho, že vývojáři a vlastně celý tým řídí sám sebe, což je ovšem podmíněno tím, že lidé v týmu musí být zodpovědní a spolehliví. Metodika podporuje intenzivní komunikaci v týmu i s klientem, kterou různými typy schůzek přímo vynucuje. Díky tomu v týmu i u zákazníka obvykle panuje velmi dobrá informovanost a povědomí o celém projektu [23, s. 18-22]. Cílem Scrumu je dosažení vysoké efektivity práce projektového týmu, které chce dosáhnout při aplikaci minima omezení a současném zachování motivovaného a spokojeného týmu. Na druhé straně zachování tak vysoké úrovně nezávislosti na vedení si lze dovolit pouze u zkušených týmů, které dokáží řídit samy sebe. Scrum je tudíž velmi nevhodný pro týmy bez předešlých zkušeností. Dalším omezením plynoucím z nezávislosti týmu je maximální počet členů. Z hlediska schopnosti týmu uřídit sám sebe a vzhledem ke konání denních schůzek není možné, aby projektový tým disponoval velkým množstvím členů. Další výhody a nevýhody plynou obecně z agilního přístupu, kterému jsme se již věnovali.

5.2 Lean Development

Lean Development se z hlediska dělení metodik a projektových přístupů někdy řadí do samostatné kategorie po boku tradičního a agilního přístupu, který nese název Štíhlé projektové řízení, anglicky Lean Project Management [29] [30]. Někteří autoři štíhlý přístup považují za samostatnou kategorii z hlediska dělení přístupů, jiní jej považují za součást agilního přístupu. Například Wysocki jde ve své knize ještě o krok dál a obě tyto myšlenky kombinuje, pojednává o Štíhlém agilním přístupu, který má svá vlastní specifika [6, s. 312]. Nicméně my v této diplomové práci budeme, stejně jako mnoho dalších autorů, vycházet z předpokladu, že Lean Development je metodikou agilní. Proč tomu tak je, si detailněji vysvětlíme dále v této kapitole.

Zajímavostí této metodiky je její původ, který pochází z oblasti strojírenství, konkrétně z automobilového průmyslu. Inspirací byl výrobní proces japonské automobilové značky Toyota s názvem Toyota Production System, dnes nazývaným také jako Štíhlá výroba, anglicky Lean Manufacturing [10, s. 158]. Důležité jsou principy, které Toyota aplikovala na jejich výrobu automobilů, jelikož tyto principy s patřičnými úpravami lze aplikovat i na další procesy nad rámec hromadné výroby, třeba na projektové řízení a vývoj jakéhokoliv produktu. Štíhlá výroba, respektive princip štíhlosti, spočívá v eliminaci všeho zbytečného, co zapříčiňuje zbytečné navyšování nákladů a snižování efektivity procesu [10, s. 158]. Jinými slovy, cokoliv po nás zákazník explicitně nevyžaduje, nebo co mu nepřináší přidanou hodnotu, je z procesu odstraněno. Toyota tímto způsobem dosáhla výrazného navýšení efektivity v oblasti výroby, avšak za povšimnutí stojí i výsledky v oblasti samotného procesu vývoje. V článku Lean Development od pánů Freddyho a Michaela Ballé je uvedeno, že v Toyotě mohou projekty týkající se produktového vývoje dosahovat poloviční délky s čtyřnásobnou produktivitou v porovnání s americkou konkurencí [31]. Autoři na základě reportu z National Center for Manufacturing Sciences[5] uvádí i konkrétní čísla, zatímco Chrysler potřebuje na vývoj nového modelu cca 600 inženýrů, Toyota dosáhne obdobného výsledku za poloviční dobu s nasazením pouhých 150 inženýrů.

5.2.1 Základní charakteristika

Metodika Lean Development se dá popsat jako souhrn pravidel vedoucí k objevení a eliminování potenciálních zdrojů plýtvání během celého procesu vývoje produktu, čímž je dosaženo navýšení efektivity procesů a snížení nákladů. Na rozdíl od ostatních metodik nenabízí žádné konkrétní nástroje, ale pouze popisuje obecné principy, jejichž následováním je zvýšení efektivity docíleno. Lean Development si dává neskromné cíle v podobě úspěšného vývoje produktu v třetinovém čase, s třetinovými náklady a s třetinovou chybovostí [10, s. 158]. Stěžejní myšlenkou celé metodiky je maximalizace zákazníkovy přidané hodnoty při současné minimalizaci plýtvání. Za tímto účelem jsou definovány klíčové principy, jejichž aplikace i formulace se mohou s ohledem na typ projektu lišit, jiné principy lze aplikovat na softwarový projekt a jiné zas na průmyslovou výrobu, avšak podstata štíhlého myšlení zůstává pořád stejná [32]. Z různých zdrojů lze tedy dohledat odlišné formulace principů štíhlého přístupu.

[5] National Center for Manufacturing Sciences je americká nezisková organizace sdružující společnosti za účelem vývoje nových technologií napříč průmyslovými odvětvími.

5.2.2 Sedm principů metodiky Lean Development

V této kvalifikační práci budeme čerpat z formulace, jejíž autory jsou Marry a Tom Poppendieck, kteří ve své knize „Lean Software Development: An Agile Toolkit“ specifikují 7 principů, na nichž Lean Development stojí [33]. Je důležité si uvědomit, že autoři vychází z předpokladu, že metodika bude aplikována na vývoj softwaru, respektive na vývoj nového produktu, z tohoto předpokladu budeme vycházet i my. Avšak kdyby se jednalo o projekt týkající se výroby či kompletace produktu již navrženého, např. realizace developerského projektu, pak bychom museli zahrnout i některé principy vycházející z původní štíhlé výroby týkající se minimálních zásob, resp. just-in-time zásobování, partnerství s dodavateli atd. V případě vývoje softwaru a vývojových projektů obecně lze principy Lean Development formulovat následovně:

- **Odstranění plýtvání** – jakýkoliv druh plýtvání je krajně nechtěný. Z hlediska uspokojení zákazníka je důležité soustředit se pouze a jen na uspokojení zákazníkových požadavků a jakékoliv prostoje či meziprodukty, které jsou nad rámec požadavků, nebo nejsou nezbytně nutné k efektivnímu splnění zadání, musejí být vyhledány a eliminovány.
- **Podpora neustálé optimalizace a učení v týmu** – jelikož v projektech týkajících se vývoje nového produktu se často setkáváme s nekonvenční problematikou, je nezbytné lpět na neustálém vzdělávání týmu a snaze o zdokonalování procesů. Důležitou součástí agilního řízení projektů je iterativní přístup, přičemž každá iterace slouží týmu jako drahocenná zkušenost do iterací následujících. Tento proces obohacování projektového týmu a procesů je nezbytné rozvíjet, není-li tomu tak, dá se hovořit o plýtvání potenciálem projektového týmu.
- **Rozhodnutí v nejpozdnější možné fázi** – pro projekt je velmi přínosné, jsou-li rozhodnutí uskutečněna spíše později nežli dříve. V takovém případě je problematika daného rozhodnutí aktuální a podložena fakty z aktuální situace, nikoliv spekulacemi jako v případě, že k těmto rozhodnutím dochází s velkým časovým předstihem. Pozdější rozhodování je úzce spjato také s principy samotného agilního přístupu, který vyzdvihuje riziko změn a potřebu je implementovat, přičemž odložením některých rozhodnutí se dá množství změnových požadavků přímo předejít.

- **Dodávky v nejnižších možných intervalech** – rychlost vývoje je klíčová z hlediska zpětné vazby a všech předešlých principů, které jsme si u této metodiky zatím uvedli. Rychlý vývoj a krátké iterace umožňují zákazníkovi podávat častou zpětnou vazbu a diktovat si požadavky, které aktuálně potřebuje. Tímto je umožněno klientům odložit některá rozhodnutí na pozdější vývojové cykly, kdy budou aktuální a kdy k nim budou mít náležité podklady z předešlých iterací. Rychlý vývoj slouží týmu také k lepšímu vzdělání, chápeme-li vzdělávací proces jako posloupnost návrhu, implementace a zpětné vazby, na jejímž základě dochází ke vzdělání týmu a zdokonalování procesů. V neposlední řadě je zmenšení procesu výroby přidané hodnoty jedním ze základních nástrojů štíhlé výroby pro odstranění plýtvání.
- **Zplnomocnění týmu** – kvalita závisí na detailech a detaily závisí na samotném projektovém týmu. Je v zájmu projektu poskytnout projektovým členům určitou úroveň rozhodovacích pravomocí v technických a procesních otázkách implementace. Zkušený projektový pracovník pod vedením projektového manažera je schopen lépe vyhodnotit situaci v konkrétním projektu než centrální autorita, tím pádem by měl být i tím, kdo bude vykonávat rozhodnutí ohledně technického řešení. Zplnomocnění pracovníků dále slouží jako motivační prostředek, jelikož každý pracovník je za svá rozhodnutí plně zodpovědný, což jej motivuje ke kvalitně odvedené práci.
- **Zavedení integrity** – integrita je důležitá z hlediska produktu i vývojového procesu. I přes snahu eliminovat jakékoliv plýtvání musí být dodržena celistvost vývojového procesu, tak aby se projektový tým mohl zaručit za maximální kvalitu výsledného produktu. Integrita procesu vývoje je hlavním předpokladem, aby výsledný produkt vyhovoval zákazníkovým představám v plném rozsahu.
- **Vidět celek** – tento princip přímo souvisí s integritou produktu a optimalizací dílčích částí. Lean Development považuje optimalizace dílčích částí za plýtvání. Je nezbytné dívat se na projekt jako na celek a na základě požadavků nastavit všechny dílčí části tak, aby spolu adekvátně spolupracovaly. Lokální optimalizace nad rámec projektových požadavků jsou nepotřebné a tudíž nechtěné.

5.2.3 Hodnocení metodiky Lean Development

Metodika Lean Development je především o „štíhlém“ myšlení a přístupu, jež lze obecně aplikovat na jakýkoliv proces, projektové řízení tedy není výjimkou. Podíváme-li se na výše uvedené principy, zjistíme, že v nich lze nalézt analogii s agilním manifestem. Stejně jako ostatní agilní metodiky i Lean Development vyzdvihuje přidanou hodnotu pro zákazníka, důležitost implementace změn a klíčovou roli projektového týmu i samotného klienta. Lean Development navíc k agilním zásadám přidává snahu o optimalizaci procesu z hlediska eliminace jakéhokoliv plýtvání, což se projevuje ve snížení nákladů a zkrácení délky celkového procesu.

Zaměříme-li se konkrétně na samotnou metodiku, ta zahrnuje spíše obecné principy, kterými se řídit, aby bylo dosaženo vytyčených cílů, přičemž konkrétní nástroje a návod, jak tyto principy aplikovat, již nenabízí [10, s. 176]. Dalo by se tedy říci, že metodika cílí spíše na strategickou úroveň, zatímco ostatní agilní metodiky se zaměřují obvykle na úroveň taktickou [19, s. 51].

5.3 Dynamic Systems Development Method

Dynamic Systems Development Method (DSDM) je další z řady agilních metodik, které byly původně navrženy pro projekty týkající se vývoje softwaru. Nicméně metodika se od jejího vzniku v roce 1994 neustále vyvíjí a pozdější verze jsou navrženy tak, aby s jejich pomocí mohly být řízeny i ostatní projekty, jež se netýkají pouze informačních technologií či vývoje softwaru [34].

Metodiku vyvinulo DSDM konsorcium, které se dodnes stará o její vývoj. Vznik tohoto neziskového uskupení se datuje do roku 1984, kdy se různé organizace z veřejného i soukromého sektoru spojily za účelem řešení, v té době, nákladného a zkostnatělého softwarového vývoje. Jejich společné snažení vyvrcholilo vznikem metodiky DSDM a později se konsorcium podílelo i na tvorbě Agilního manifestu vývoje softwaru. Metodika DSDM prošla od jejího vzniku dlouhým vývojem a množstvím verzí, které reagovaly na měnící se podmínky ve světě projektového řízení. Nejaktuálnějšími verzemi jsou DSDM Atern z roku 2008 a DSDM Agile Project Framework z roku 2014 [34].

5.3.1 Základní charakteristika

Metodika DSDM vychází z přístupu s názvem Rapid Application Development (RAD), která klade velký důraz na samotný proces vývoje, především pak na délku cyklu vývoje prototypů, a menší důraz na detailní plánování [19, s. 45]. DSDM obdobně jako RAD využívá prostředků prototypování nebo iterativního a inkrementálního přístupu, které kombinuje s dalšími agilními principy. Jedná se o univerzální metodiku, která je vhodná pro širokou škálu různých projektů. Lze ji aplikovat na malé i velké projekty zabývající se různou problematikou od vývoje softwaru až po optimalizaci podnikových procesů [35]. Z hlediska obsahu metodika definuje klíčové principy a hodnoty, procesní strukturu projektu, role s jejich povinnostmi, konkrétní nástroje atd. Jedná se tedy o metodiku relativně obsáhlou, která nabízí konkrétní nástroje a postupy, jak dané klíčové principy aplikovat v praxi, přičemž metodika řeší problematiku celého životního cyklu projektového řízení od počáteční fáze tvorby hrubého konceptu až po finální předání výsledného produktu a oficiální ukončení projektu.

DSDM vychází z předpokladu, že projekt by měl disponovat tzv. dostačujícím předběžným plánem, čímž se liší od ostatních metodik, jež v případě tradičního přístupu praktikují detailní plán, a v případě některých agilních metodik může být plánování téměř vypuštěno. Tato metodika tedy umožňuje integrovat agilitu do procesu řízení projektu při zachování určitého stupně předběžného plánování, jehož absence je trnem v oku některým organizacím [35]. Z hlediska projektového řízení a fungování metodiky jsou pak důležité především základní principy, ze kterých DSDM vychází.

5.3.2 Osm základních principů

Původně metodika DSDM specifikovala 9 principů, se kterými se lze ještě v některé literatuře setkat, nicméně s vývojem metodiky se principy reformulovaly, což mělo za následek snížení jejich celkového počtu. Dnes se uvádí 8 stěžejních principů, které jsou prezentovány v aktuální verzi z roku 2014 s názvem DSDM Agile Project Framework. Oněch 8 principů pak lze na základě oficiální dokumentace metodiky prezentovat následovně [35]:

1. **Tvorba přidané hodnoty** – Hlavním cílem projektu je přinést zákazníkovi určitou přidanou hodnotu. Jinými slovy důležité je dodat zákazníkovi to, co potřebuje a

v čase, v jakém to potřebuje. Z tohoto důvodu je důležitá aktivní účast zákazníka na projektu a správné porozumění dané projektové problematiky ze strany projektového týmu. Zákazníkův prospěch by měl plynout ze všech rozhodnutí vykonaných po dobu projektu, za tímto účelem DSDM zavádí metody s názvem Timeboxing a MoSCoW Prioritisation.

- **Timeboxing** se dá přirovnat k tvorbě Sprintů ve Scrumu. Jedná se o iterace s jasně definovanými cíli, fixní délkou (obvykle 2-4 týdny) a jasně stanoveným datem, do kdy musí být ony příslušné cíle dosaženy.
- **MoSCoW Prioritisation** je metoda sloužící ke specifikaci priority cílů v rámci iterace. Pomocí této metody projektový tým rozdělí potenciální cíle do 4 kategorií od nejdůležitějších s nejvyšší prioritou po ty nejméně důležité, které v rámci iterace implementovány nebudou. Ony kategorie jsou v původním anglickém znění následující: Must have; Should have; Could have; Won't have.

2. **Včasné dodávky –** Je-li zavedený inkrementální iterativní přístup k projektovému řízení, je třeba dodržovat stanovené termíny a dodávat jednotlivé inkrementy včas. Není-li tomu tak, zákazník může začít nabývat dojmu, že projekt není dostatečně pod kontrolou. V případech, kdy projekt cílí na určitou podnikatelskou příležitost nebo je součástí projektu zákonná lhůta, pak může mít jakékoliv zpoždění fatální následky. Za účelem včasných dodávek DSDM zavádí výše zmíněné metody na určení časového rámce a priorit jednotlivých požadavků, které napomáhají projektovému týmu plnit cíle v rámci vytyčených termínů.
3. **Spolupráce –** Jak již bylo mnohokrát řečeno, tým, který aktivně spolupracuje, dosahuje větší efektivity v porovnání se situací, kdy každý z týmu pracuje samostatně. DSDM za účelem zvýšení efektivity práce kombinuje v projektovém týmu lidi s technickým i odborným zaměřením a vyzývá k aktivnímu zapojení kompetentních lidí ze strany zákazníka během projektu. Každý člen projektového týmu je oprávněn vykonávat rozhodnutí v oblasti jeho expertízy, která jsou v nejlepším zákazníkově zájmu a za která je daný pracovník zodpovědný. DSDM definuje i konkrétní nástroj s názvem Facilitated Workshops, který z tohoto principu vychází.

- **Facilitated Workshops** jsou pracovní setkání, která sledují nějaký konkrétní cíl a kterých se účastní pouze lidé (role), kteří jsou k řešení dané problematiky vybráni a jejichž přítomnost na setkání je přínosná. Samotné setkání pak organizuje a řídí speciální role s názvem Workshop Facilitator. Tyto schůzky slouží k rychlému a kvalifikovanému rozhodování, zapojení všech zainteresovaných stran, budování týmového ducha a ujasnění probírané problematiky [10, s. 55].

4. **Zachování kvality –** Kvalita je otázkou specifikace a neměla by vytvářet v rámci projektu proměnnou. Před zahájením vývoje produktu je nezbytné domluvit úroveň kvality, která zůstane zachována po celou dobu projektu. Produkt odevzdaný v domluvené kvalitě je pak považován za dostatečně kvalitní a projekt za úspěšný. Nástroji kvality je adekvátní návrh a dokumentace, kontroly a průběžné testování, které je integrováno do samotného procesu vývoje. V případě IT projektů se dá po boku DSDM komplementárně využít Vývoj řízený testy (TDD) či jiné metodiky vývoje softwaru.
5. **Inkrementální vývoj na pevných základech -** Pevnými základy se v tomto případě rozumí dostatečný návrh a plán z počáteční fáze projektu. Předpokladem je, že vývoji by měla předcházet nějaká zevrubnější analýza a porozumění při zachování úrovně agility. Nejedná se tedy o detailní analýzu, jako tomu je v případě Vodopádového modelu, kde podrobné plánování způsobuje celkovou zkostnatělost. Samotný produkt je pak vyvíjen inkrementálně, což umožňuje přinášet zákazníkovi přidanou hodnotu postupně a v čase, kdy ji potřebuje. Zákazník zároveň může týmu zpětně poskytovat zpětnou vazbu z již implementovaných částí. Tento princip je do projektového řízení integrován skrze definovaný životní cyklus projektu podle metodiky DSDM.
6. **Iterativní vývoj –** S inkrementálním vývojem přímo souvisí i iterativní vývoj. Dodávané inkrementy jsou vypracovány v rámci iterací, v DSDM označovaných Timeboxy, díky čemuž je možné lépe integrovat změny do procesu vývoje a dochází k častější zpětné vazbě. Iterativní vývoj slouží jako prostředek k experimentování, vzdělávání a zdokonalování projektového týmu, a umožňuje ladění produktu, jehož první zpracování může v zákazníkovi vyvolat touhu po změně a optimalizaci. Změny jsou v rámci omezení času a nákladů vítané, jelikož

právě změny jsou prostředkem k tvorbě přidané hodnoty a celkové spokojenosti zákazníka.

7. **Aktivní komunikace** – Nedostatečná komunikace bývá nejčastějším důvodem selhání projektu, proto DSDM přikládá této problematice velkou váhu a vyzdvihuje důležitost frekventované, upřímné a transparentní komunikace v týmu i mimo něj. Konkrétními nástroji, kterých DSDM využívá za účelem dosažení efektivní komunikace, je neformální a osobní komunikace napříč projektem, konání Daily Stand-up schůzek (obdoba Daily Scrum Meetingu), vizuální prostředky jako modelování či prototypování, a v neposlední řadě adekvátní úroveň dokumentace.
8. **Kontrola nad projektem** – Je nezbytné mít projekt pevně v rukou a zajistit u všech projektových procesů transparentnost, aby bylo na první pohled zřetelné, zdali je vývoj pozadu, či nikoliv z hlediska délky trvání Timeboxu a zbývající práce (obdoba Burndown diagramu). Projektoví manažeři a vedoucí týmů musejí k řízení přistupovat proaktivně a měřit aktuální úroveň dosažení vytyčených cílů, přičemž se jedná o informaci, která by měla být dostupná všem zúčastněným projektovým pracovníkům.

Následování těchto 8 principů přináší do projektového řízení agilitu a přispívá k následování filosofie metodiky, která zní:

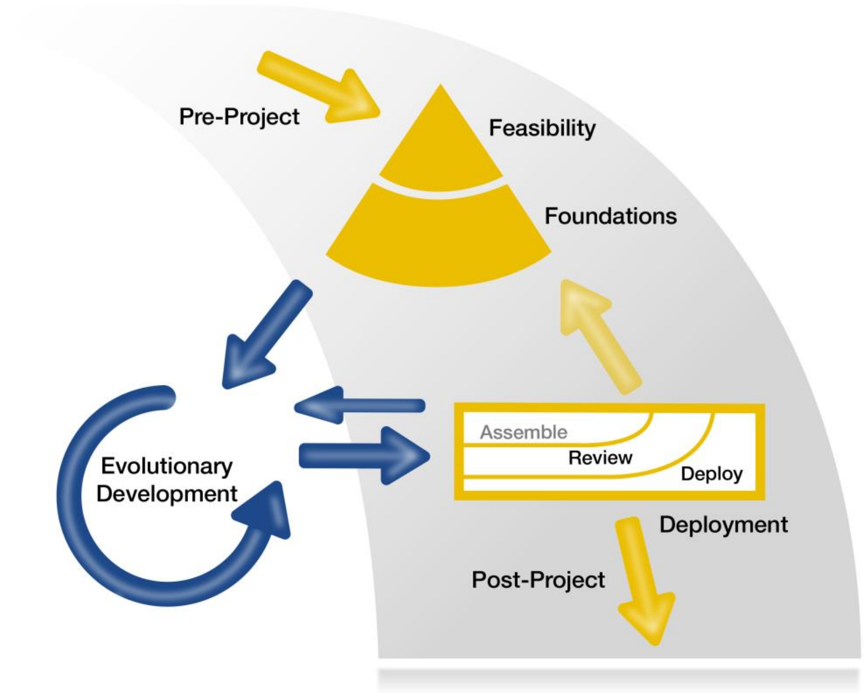
„Nejlepší přidané hodnoty pro zákazníka je dosaženo u projektů, jež sledují jasně specifikovaný cíl, mají časté dodávky a jsou realizovány za spolupráce motivovaných a zplnomocněných lidí.“ [35]

Naopak distancování projektového týmu od některého z těchto principů by zapříčinilo nenaplnění potenciálu metodiky a odporovalo by výše uvedené filosofii [35].

5.3.3 Procesní model

Metodika využívá k řízení projektu jasně definovaný procesní model, který určuje životní cyklus projektů řízených s využitím DSDM, viz Obrázek 7. Tento model sestává z 6 fází, které na rozdíl od některých ostatních agilních metodik pokrývají celý životní cyklus projektu, nikoliv pouze část zabývající se vývojem řešení [36]. Bližší představení tohoto procesního modelu bude obsahem následujícího textu.

Ještě předtím než přejdeme k jednotlivým fázím životního cyklu DSDM, je důležité podotknout, že DSDM je metodika, již je možné integrovat po boku dalších metodik či standardů projektového řízení za účelem komplementárního efektu. V praxi se lze setkat s využitím DSDM po boku dobře známých agilních metodik jako jsou Scrum a Extrémní programování. DSDM lze také úspěšně integrovat s projektovými standardy, jako jsou PRINCE2 či PMBOK [35].



Obrázek 7: Životní cyklus projektu dle metodiky DSDM
Zdroj: **[35]**

Jak již bylo řečeno, procesní model specifikuje 6 fází, přičemž stěžejní z hlediska projektového řízení jsou 4 z nich. Konkrétně se jedná o fázi proveditelnosti, specifikaci základů, fázi vývoje a fázi nasazení. Tyto fáze jsou ohraničeny zbylými dvěma, které řeší dění před zahájením a po skončení projektu.

- **Předprojektová fáze (Pre-project Phase)** má především selektivní účel, kdy je cílem vyřadit projekty, jež nekorespondují s filosofií a principy DSDM nebo postrádají jasně specifikovaný cíl. Jinými slovy účelem této fáze je zajistit, že následující fáze jsou započaty pouze u správných projektů, kde je vidina budoucího úspěchu.

- **Fáze proveditelnosti (Feasibility Phase)** je časový prostor, v rámci kterého se uskutečňuje tzv. studie proveditelnosti. Cílem této studie je podání bližších informací ohledně proveditelnosti projektu z technického i ekonomického hlediska, které se zabývá finanční náročností projektu a jeho návratností. Tato studie by měla umožnit rozhodnutí, zdali projekt realizovat, nebo zdali jej včas ukončit, protože jeho realizace není z nějakého důvodu perspektivní.
- **Specifikace základů (Foundations Phase)** je část projektu, kdy projektový tým i zákazník mají za úkol definovat základní podobu projektu, požadavky, první návrhy technického řešení atd. Tato fáze slouží k nabytí základního vědomí o projektu a detailnějšímu seznámení týmu s projektovou problematikou z technického, procesního i obchodního hlediska. Cílem není detailní analýza, ta je ponechána na proces samotného vývoje, i proto by tato fáze neměla trvat déle než pár týdnu, a to i u komplexních a velmi složitých projektů.

  Dojde-li v pozdějších fázích projektu k zásadním změnám ohledně specifikace, je možné a někdy i nezbytné se vrátit k této fázi a onu specifikaci aktualizovat tak, aby odpovídala reálnému stavu. Tato skutečnost je na Obrázku 7 zobrazena jako šipka od fáze nasazení zpět k fázi specifikace základů.

- **Fáze vývoje (Evolutionary Development Phase)** tvoří majoritní část celého projektového životního cyklu, během které je vytvářena přidaná hodnota pro zákazníka inkrementální a iterativní formou. Za účelem vývoje jsou využívány konkrétní nástroje, které metodika DSDM definuje. Práce během této fáze zahrnují také detailní analýzy, které zajistí, že finální produkt i každý inkrement odpovídá zákazníkovým představám a je správně technicky implementován.
- **Fáze nasazení (Deployment Phase)** slouží k nasazení vytvořeného produktu k ostrému provozu. Jak je vidět na Obrázku 7, nasazení lze dělit na 3 dílčí činnosti:
  - **Kompletace (Assemble)** – v případě nasazení komplexního produktu, na jehož vývoji se podílí několik jednotlivých projektových týmů, je nutné zkompletovat všechny inkrementy tak, aby tvořily celistvý produkt.
  - **Přezkoumání (Review)** – jakmile je produkt zkompletován, musí jeho nasazení předcházet nějaká forma kontroly, která může nabývat formálního i neformálního rázu. Za tímto účelem dochází po kompletaci

k přezkoumání, jenž dá podněty k nasazení produktu nebo produkt vrátí s připomínkami k nápravě. Zároveň během procesu přezkoumání dochází u projektového týmu k retrospektivě, která slouží jako kritické ohlédnutí za projektem a hledání potenciálních příležitostí k optimalizaci.

  - **Nasazení (Deploy) –** je-li produkt v rámci přezkoumání schválen, nasazení slouží k uvedení produktu do praxe, k čemuž může dojít v rámci projektu, nebo se realizace nasazení vloží do rukou samotného zákazníka.

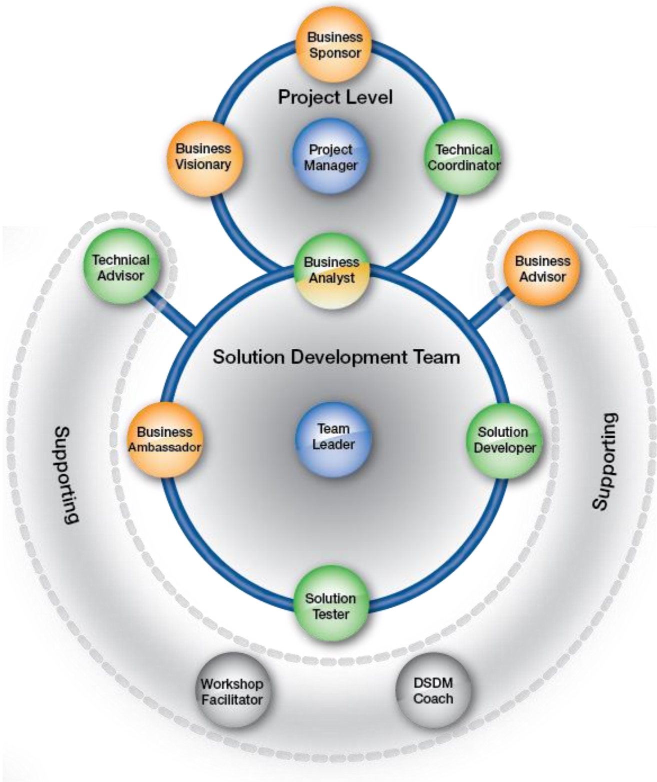
- **Poprojektová fáze (Post-project Phase)** si dává za cíl zhodnotit projekt, konkrétně jak úspěšně se podařilo splnit zákazníkovy požadavky. K tomuto hodnocení dochází po uplynutí stanovené doby, během které je produkt používán v ostrém provozu.

DSDM je metodika určená pro různé typy projektů lišících se z hlediska velikosti i úrovně formality. Na základě konkrétních vlastností daného projektu pak lze tento procesní model konfigurovat a upravovat, aby dosahoval maximální efektivity [37].

5.3.4 Role a Projektový tým

Metodika DSDM obdobně jako Scrum definuje konkrétní projektové role, přičemž u každé role podrobněji popisuje její účel a z toho vyplývající zodpovědnosti. Všechny role pak velmi úzce spolupracují na společném cíli, kterým je úspěšné dokončení projektu. Z hlediska této spolupráce je důležitá aktivní komunikace, respekt a odvaha zlepšovat sebe i spolupráci s ostatními.

Konkrétní role a jejich členění je možné vidět na Obrázku 8, který znázorňuje schéma DSDM projektovému týmu. Na první pohled je zřejmé, že metodika specifikuje velké množství různých rolí, rozdíl je patrný především v porovnání s ostatními agilními metodikami vývoje softwaru. Důvodem je fakt, že DSDM řeší otázku celého projektu, což zahrnuje samotné projektové řízení i vývoj produktu [37]. Ostatní agilní metodiky se většinou soustřeďují především na vývojovou část projektu a projektové řízení už do hloubky neřeší [36]. Při bližším pohledu na schéma zjistíme, že role jsou členěny do tří kategorií, které zastřešují role na základě jejich obecnějšímu účelu – Řízení projektu, Vývoj řešení a Podpůrné role.



Obrázek 8: Organizační struktura dle metodiky DSDM
Zdroj: **[35]**

Z hlediska sledovaných zájmů pak můžeme role dělit ještě následovně [35]:

- Role označené **modrou** barvou mají zájem na řízení projektu a projektového týmu.
    - **Project Manager** – má na starosti projektové řízení, alokaci zdrojů, správu rizik a měření vývoje projektu.
    - **Team Leader** – řídí denní aktivity vývojového týmu, motivuje tým a zajišťuje včasnou dodávku jednotlivých inkrementů.
- Role označené **oranžovou** barvou sledují především obchodní zájmy. Jedná se převážně o role reprezentující stranu zákazníka.
    - **Business Sponsor** – nejvyšší role reprezentující zákazníka, která má na starosti financování projektu a zplnomocnění ostatních rolí reprezentujících obchodní zájmy.
    - **Business Ambassador** – člen vývojového týmu reprezentující obchodní stránku řešení. Podílí se na denních aktivitách vývojového týmu, kde poskytuje poradenství a zákazníkův pohled na problematiku.

- **Business Visionary** – určuje obchodní vizi projektu a sleduje vývoj projektu oproti dané vizi. Podílí se na definici klíčových požadavků.
- **Business Advisor** – poskytuje odborný pohled na věc, obvykle zastupuje koncového uživatele.
- **Business Analyst** – zprostředkovává komunikaci a tvoří pomyslnou hranici mezi rolemi zabývajícími se obchodní a technickou stránkou, také proto je označen oběma barvami v rámci schématu.

- Role označené **zelenou** barvou se zabývají technickým řešením projektu.
  - **Technical Coordinator** – zastřešuje vývoj produktu po technické stránce, schvaluje architekturu řešení a koordinuje projektové týmy.
  - **Technical Advisor** – specialista po technické stránce, obvykle zastupuje personál, který má na starosti koncové technické řešení a jeho údržbu.
  - **Solution Developer** – klasický vývojář, podílí se na tvorbě přidané hodnoty pro zákazníka.
  - **Solution Tester** – zastřešuje testování vyprodukovaného řešení.
- Role označené **šedivou** barvou se blíže zabývají procesním hlediskem a aplikací DSDM principů.
  - **Workshop Facilitator** – neutrální role, která zajištuje a řídí semináře (Facilitated Workshops), což jsou pracovní schůze, jichž se účastní pouze vybrané a kompetentní role a jež sledují nějaký konkrétní cíl.
  - **DSDM Coach** – specialista přes metodiku DSDM, úkolem této role je poskytovat poradenství ohledně DSDM a je-li potřeba, nakonfigurovat DSDM tak, aby adekvátně odpovídal potřebám konkrétního projektu.

DSDM kategorizuje role na základě jejich účelu nebo zájmu, který sledují. Celá hierarchie rolí a jejich dělení je velmi přehledně vyobrazeno na Obrázku 7. Jedná se o formální role, které metodika definuje, nicméně platí, že jedna role může být reprezentována více lidmi, a naopak jeden člověk může reprezentovat několik rolí [35].

DSDM neudává optimální velikost projektového týmu a cílí na celý projektový tým, nikoliv pouze na vývojový tým. Úroveň řízení v týmu, především toho vývojového, pak závisí na konkrétních projektech, přičemž Team Leader může tým řídit direktivně nebo jej vést k tomu, aby řídil sám sebe, jako tomu je například u Scrumu [35]. Metodika zahrnuje a vyzdvihuje důležitost rolí zastupujících zákazníka, který se stává regulérní součástí projektu a tudíž i samotného projektového týmu. Spolupráce mezi lidmi zastupujícími zákazníka a ostatními, jež zaštiťují projektový vývoj a řízení, je zcela esenciální z hlediska budoucího úspěchu projektu.

5.3.5 Hodnocení metodiky DSDM

DSDM je velmi obsáhlou metodikou, která se zaměřuje na komplexní problematiku celého projektového řízení a která nabízí konkrétní nástroje, kterých při řízení projektů využívá. Metodika se neustále vyvíjí a její poslední verze se datuje do roku 2014, což lze považovat za velkou výhodu oproti ostatním agilním metodikám, které již aktivnímu vývoji nepodléhají. Díky tomuto vývoji lze dnes DSDM použít na různé typy projektů, které se zabývají různou problematikou, na rozdíl od původních verzí, které cílily především na vývoj softwaru. V mnoha ohledech, především pak u konkrétních nástrojů jako Timeboxing a Stand-up meeting, lze nalézt analogii s metodikou Scrum.

5.4 Extrémní programování

Extrémní programování (XP) je metodika zaměřená výhradně na projekty týkající se vývoje softwaru a horko těžko bychom hledali její uplatnění i v jiných oblastech, čímž se významně odlišuje od ostatních agilních metodik, které jsme tu již zmínili. Nicméně jedná se o agilní metodiku natolik populární a významnou, že stojí za to o ní alespoň stručně pojednat.

Autorem metodiky je Kent Beck, který je zároveň jedním z prvních signatářů Agilního manifestu vývoje softwaru. Beck se v rámci metodiky zaměřuje především na vývojovou fázi projektu a formální stránku projektového řízení opomíjí, přičemž tuto část nechává volně na rozhodnutí projektového týmu [10, s. 138].

5.4.1 Základní charakteristika

Extrémní programování využívá osvědčené postupy, které jsou běžně využívány při vývoji softwaru, avšak dovádí je do extrému [19, s. 57]. Jedná se o metodiku účinnou, lehkou a flexibilní, která je postavena na jasně definovaných principech a umožňuje efektivní řešení projektů s nejasnými či proměnlivými požadavky. XP je určeno především pro menší týmy, ve kterých se počet programátorů pohybuje v jednotkách. Jak již bylo uvedeno, XP dovádí osvědčené principy a postupy do extrému. Ony principy lze pak popsat následovně [10, s. 126] [19, s. 57]:

- **Jednoduchost**, za předpokladu splnění požadavků, je žádoucí vlastností jakéhokoliv řešení. Cílem je tedy vyvíjet a dodávat to nejjednodušší možné řešení, které je schopno fungovat podle očekávání zákazníka.
- **Revize a kontroly** jsou prováděny nepřetržitě během vývoje, čehož je dosaženo formou tzv. párového programování[6].
- **Návrh** je neustále upravován a vylepšován. Dochází k tzv. refaktorizaci, kdy se upravuje zdrojový kód, přičemž jeho funkce se nemění.
- **Architektura** je důležitou součástí každého softwaru a důležitým faktorem z hlediska úspěchu či neúspěchu projektu. V XP je proto vytvořena tzv. metafora, která slouží jako příběh, pomocí kterého lze ohledně architektury komunikovat bez složitého a vyčerpávajícího zasvěcení nových lidí do technických a procesních detailů projektu.
- **Testování** je neustále prováděno ze strany vývojářů, kteří testují jednotlivé jednotky a třídy, i ze strany zákazníků, jež testují funkcionality.
- **Integrace** a její testování je u větších projektů nedílnou součástí vývoje softwaru, proto integrace i jejich testování probíhají několikrát denně.
- **Krátké iterace** se v dnešní době osvědčily jako efektivní prostředek k častějšímu testování a užší spolupráci se zákazníkem. XP přistupuje extrémně i k délce iterací, které mohou být v řádech hodin či dnů.

XP dále vyzdvihuje klasické agilní hodnoty jako je týmová práce, komunikace nad dokumentací či spolupráce se zákazníkem, který je nedílnou součástí vývoje softwaru. Důležitou součástí metodiky tvoří kromě výše uvedených postupů také 5 základních hodnot, které jsou považovány za stěžejní z hlediska úspěšné aplikace XP. Jedná se o komunikaci, jednoduchost, zpětnou vazbu, vzájemné respektování všech členů týmu a odvahu změnit již existující řešení ve prospěch finálního produktu [38].

[6] Párové programování je způsob programování, při kterém dva programátoři pracují společně na jednom zdrojovém kódu. Oba programátoři pak zpravidla sdílejí jeden počítač a společnými silami vytvářejí zdrojový kód, přičemž efektivita zůstává přinejmenším stejná jako u dvou programátorů, jež pracují samostatně, avšak kvalita výrazně roste a chybovost klesá [38].

5.4.2 Hodnocení metodiky Extrémní Programování

XP je postaveno na řadě jednoduchých pravidel, ovšem jeho aplikace a osvojení v reálném světě už tak prosté nejsou. Z hlediska vývoje softwaru je tato metodika velice nekonvenční a to může činit některým lidem velké problémy. Kadlec [10, s. 139] ve své knize uvádí zajímavou myšlenku ohledně určité spojitosti uplatnění této metodiky s národní mentalitou. Zatímco v USA je XP velmi populární, v České Republice se k této metodice stavějí vývojáři spíše zdráhavě. Příčinou je podle autora česká mentalita, která odmítá neúspěch a předělávání již zhotovené práce, což je z hlediska XP zásadní problém. Naopak v USA je mentalita taková, že lidé nemají obecně problém několikrát vybrat špatný postup, předtím než vyberou ten správný, a ještě celý proces nakonec prohlásit za úspěšný. Uplatnění takové metodiky pak může být podstatně komplikovanější v českých podmínkách oproti USA, kde vznikla. Nehledě na to, že spoustě lidí může párové programování, neustálá revize a refaktorizace kódu ostatními členy projektového týmu jednoduše vadit. Osobnostní typy vývojářů tedy hrají také důležitou roli v otázce úspěšného osvojení XP.

Mezi výhody metodiky lze zařadit její přirozenost a přímočarost. XP využívá především nástrojů a principů, které se u vývoje softwaru již v minulosti osvědčily, a pouze tyto principy umocňuje. Intenzivní týmová práce a párové programování napomáhá zlepšovat kvalifikaci vývojového týmu, snižovat chybovost a zvyšovat efektivitu práce. Nepochybnou výhodou je také fakt, že XP nelpí na formalitách a soustředí se především na výsledek, ke kterému se dostává pomocí inkrementálního iterativního přístupu.

5.5 Další agilní metodiky

Oblast agilního projektového řízení je velmi rozsáhlá, ale rámec této práce neumožňuje se detailněji zabývat všemi metodikami, které do tohoto směru náleží. Za tímto účelem si v této kapitole stručně představíme některé další agilní metodiky, jež stojí za zmínku, nicméně jejich obsáhlejší popis nebyl vzhledem k rozsahu diplomové práce možný.

5.5.1 Feature Driven Development

Feature Driven Development (FDD) je metodika opět výhradně z oblasti softwarového vývoje, která pojednává o vývoji řízeným funkcionalitami, resp. užitnými vlastnostmi (features). Z hlediska této metodiky je velmi důležité modelování, jelikož právě vytvořením celkového modelu systému začíná vývojová část projektu [10, s. 179]. Model slouží k identifikaci konkrétních dílčích částí, v tomto případě seznamu užitných vlastností, které jsou následně v rámci krátkých obvykle dvoutýdenních iterací navrženy a implementovány. Celý proces vývoje systému pak lze dle metodiky dekomponovat na následujících 5 dílčích procesů [39]:

- vytvoření celkového modelu,
- vytvoření seznamu jednotlivých funkcionalit,
- plánování podle funkcionalit,
- návrh podle funkcionalit,
- implementace podle funkcionalit.

Charakteristickým rysem FDD je členění vývojového procesu, přičemž vývoj jednotlivých funkcionalit je prováděn v rámci velmi krátkých iterací. Tento přístup má za následek časté dodávky dílčích inkrementů, které zákazníkovi umožňují dávat frekventovanou zpětnou vazbu a inkrementálně získávat finální řešení. Rozdílem oproti Scrum či XP je kromě jiného i menší autonomie samotných vývojářů, kterým je práce zadávána z vrchu [27, s. 44].

5.5.2 Test Driven Development

Test Driven Development (TDD), u nás známý především jako Vývoj řízený testy, je opět metodika určená výhradně k vývoji softwaru. Nicméně se jedná o metodiku, která v rámci vývoje softwaru zavedla zcela revoluční přístup. Testování je a vždy bylo důležitou součástí vývoje softwaru. TDD jej však nově považuje za úplný základ, který udává zbytek celého vývoje [10, s. 197]. Testy vznikají na základě specifikací ještě před samotným kódováním a právě testy slouží jako podklad pro vznikající kód. Cílem je pak napsat takový zdrojový kód, který dokáže úspěšně projít testem, nic míň, ale ani nic víc [27, s. 45].

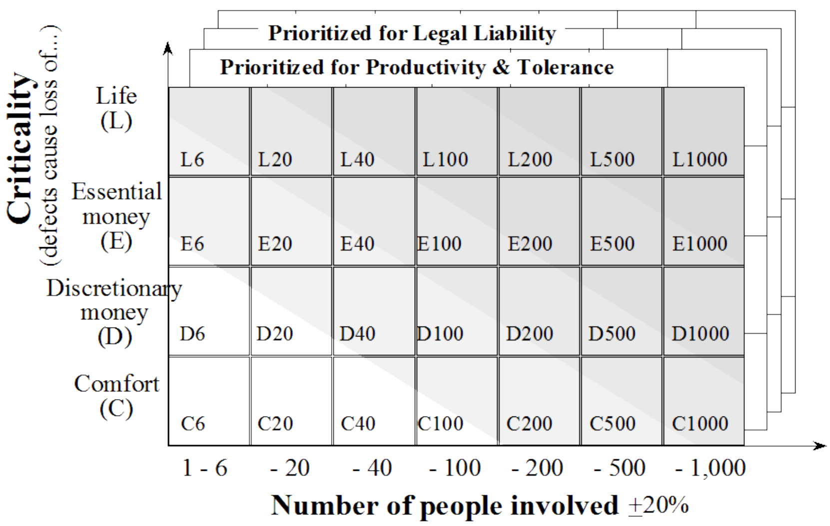
Autorem metodiky je Kent Beck, jeden z prvních signatářů Agilního manifestu softwarového vývoje, který je zároveň autorem metodiky XP. Beck formuluje ve své knize jednoduché schéma – červená, zelená, refaktorování, kterým lze jednoduše popsat princip fungování TDD v praxi [40, s. 9]:

- **Červená** znamená napsání prvotního testu, který selže, nebo jej nebude s největší pravděpodobností možné vůbec zkompilovat.
- **Zelená** znamená úspěšně zvládnutí testu za použití všech možných prostředků tak, aby test proběhl rychle a bez komplikací
- **Refaktorování** znamená závěrečnou úpravu zdrojového kódu, který je výsledkem snahy o úspěšné projití testem. Během refaktorování se nijak nemění funkcionalita kódu, pouze se odstraňují duplicity a celý kód se optimalizuje.

TDD je především metodikou vývoje softwaru, která nabízí netradiční, avšak velmi efektivní přístup k vývoji, jehož výsledkem je kvalitní a předvídatelný software. Metodika nicméně postrádá procesní hledisko nad rámec vývoje softwaru, je pak pouze na projektovém týmu, jak přistoupí k samotnému řízení projektu [10, s. 211].

5.5.3 Metodiky Crystal

Metodiky Crystal reprezentují celou rodinu metodik, jejímž autorem je Alistair Cockburn, jenž je zároveň spoluautorem Agilního manifestu softwarového vývoje [21]. Cockburn vychází z předpokladu, že není možné, aby jediná metodika vyhovovala požadavkům všech projektů, což jej vedlo k vytvoření skupiny metodik, která rozlišuje projekty na základě dvou, resp. třech, kritérií – závažnosti projektu (osa Y) a počtu zapojených lidí (osa X), viz Obrázek 9. Třetím kritériem, jež v rámci obrázku určuje posun do prostoru, je zacílení metodiky na různé priority projektu – produktivita, právní odpovědnost apod.



Obrázek 9: Schéma metodik Crystal
Zdroj: **[41]**

Na základě množství lidí, kteří na projektu pracují, jsou rozlišovány různé metodiky, které jsou označeny barevnými kódy (Clear, Yellow, Orange,…, Blue), přičemž závažnost projektu pak určuje úroveň formálních a bezpečnostních opatření, jež jsou v rámci dané metodiky uplatňovány [42]

Obecně lze říci, že metodiky Crystal kladou důraz na člověka a osobní komunikaci před procesy a dokumentací. To se v praxi projevuje snahou o maximální štíhlost procesů a minimum byrokracie, jež stále umožní efektivní řízení projektu. Metodiky vycházejí z osvědčených principů agilního přístupu, jakými jsou časté dodávky, efektivní komunikace či tendence zlepšovat sebe i samotné procesy [41]. Navíc však metodiky Crystal přinášejí do světa agilního projektového řízení revoluční škálovatelnost, která je prováděna s ohledem na charakteristiku jednotlivých projektů [19, s. 54].

5.6 Závěrem o agilních metodikách

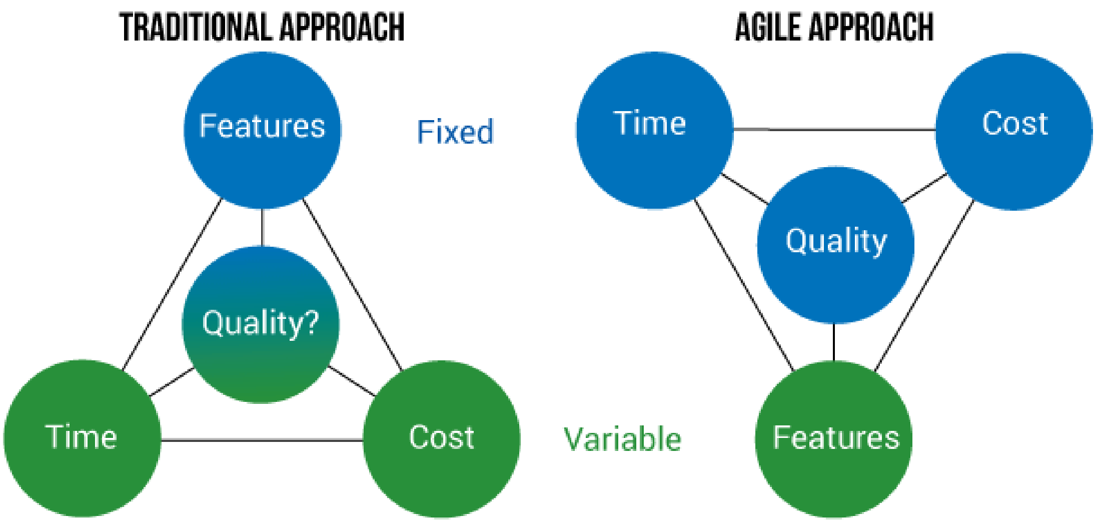
V předešlých kapitolách jsou popsány dobře známé agilní metodiky, které se dnes ve světě hojně využívají za účelem řízení projektů všeho druhu. Nicméně absolutní většina metodik zmíněných v této práci, s výjimkou štíhlého přístupu, je a byla navržena primárně pro projekty týkajících se vývoje softwaru. Metodiky lze pak částečně rozlišovat z hlediska jejich aplikovatelnosti na různé typy projektů. Zatímco Scrum a DSDM jsou komplexní agilní metodiky, které lze nakonfigurovat tak, aby je bylo možné použít na různé typy projektů, metodiky jako XP, FDD a TDD lze jen stěží aplikovat na projekty, jež se vývoje softwaru netýkají. Lean Development, resp. štíhlý přístup obecně, jakožto prostředek k omezení plýtvaní pak tvoří výjimku, kterou lze aplikovat téměř na jakýkoliv proces včetně řízení různých typů projektů.

Dalším aspektem, ve kterém se výše zmíněné metodiky liší, je jejich zaměření a účel. Některé metodiky jako Scrum a DSDM se zaobírají celou problematikou projektového řízení a řeší procesní hledisko od počátku projektu, přes fázi vývoje až po předání výsledného řešení zákazníkovi, čímž je celý projekt formálně ukončen. Tyto metodiky předepisují ucelený rámec jak celý projekt řídit i konkrétní nástroje, které za daným účelem definují [36]. Na druhé straně agilní metodiky jako XP, FDD, TDD se zabývají pouze procesem vývoje, který se na projektovém řízení podílí jen z části. Tyto metodiky pak nelze použít za účelem projektového řízení obecně, nýbrž pouze k vývoji a dodání konkrétního řešení. Je nezbytné tuto skutečnost u zmíněných metodik rozlišovat a v případě výběru metodiky pro konkrétní projekt na to pamatovat. Nicméně je-li náplní této kvalifikační práce představit si agilní metody projektového řízení, pak je zcela jistě na místě zmínit i metodiky, které se věnují pouze vývoji, konkrétně pak vývoji softwaru, jelikož právě ty často přinášejí inovativní metody do celé oblasti projektového řízení.

6 Porovnání tradičních a agilních metodik

V předchozích kapitolách jsme si stručně přiblížili různé přístupy k projektovému řízení, většina práce pak byla věnována detailnějšímu popisu konkrétních metodik z řad tradičního a především agilního přístupu. Náplní následující kapitoly bude oba tyto přístupy vzájemně porovnat a přiblížit, v čem jsou oba přístupy tak odlišné.

Na začátku této kvalifikační práce se uvádí, že tradiční přístup k projektovému řízení je využíván v situaci, kdy cíl projektu i způsob jeho dosažení jsou známé. V takovém případě lze uplatnit tradiční metodiky, které vycházejí z kvalitní specifikace požadavků a zevrubného plánu. Na druhé straně agilní přístup také vychází z předpokladu, že cíl (problém) je známý, nicméně v tomto případě již cesta k jeho dosažení (řešení) už tak jasná není [7]. V takovém případě nelze projekt předem detailně naplánovat a některá stěžejní rozhodnutí musejí být vykonána za pochodu [6, s. 48]. Za těchto okolností lze o obou přístupech říci, že tradiční projekty jsou řízené plánem a agilní projekty jsou řízené změnou, čímž se dostáváme k rozdílu v základních proměnných projektového trojimperativu, viz následující obrázek.



Obrázek 10: Porovnání tradičního a agilního přístupu z hlediska 3 základních faktorů
Zdroj: Traditional project management vs Agile [online]. [cit. 2016-11-20]. Dostupné z: https://www.quanta.co.uk/sites/default/files/docs/Cost-quality-features.png

V tradičním případě je hlavním řídícím vodítkem projektu plán, jehož součástí je i dokument s názvem Specifikace požadavků. Právě tento dokument přesně definuje, které náležitosti je nezbytné v rámci projektu zapracovat, požadavky jsou zde považovány za fixní, přičemž kvalita se odvíjí od množství času a nákladů vyčleněných pro daný projekt, které vystupují v roli variabilních [10, s. 55]. V případě agilního přístupu je tomu přesně naopak. Projektu jsou na počátku zadavatelem stanoveny vyhrazené zdroje v podobě času a nákladů, které jsou v tomto případě fixní, a jelikož jsou agilní projekty řízeny změnou, v roli proměnných vystupují požadavky, resp. funkcionalita či jiné vlastnosti produktu [10, s. 55]. Je nutné podotknout, že se jedná o základní model porovnání obou přístupů a idealizaci, která vychází ze základních principů obou přístupů. V reálné situaci se samozřejmě vyskytují výjimky, kdy jsou např. v rámci tradičních projektů měněny po domluvě se zadavatelem požadavky, nebo je u agilního projektu manipulováno s vyhraněnými zdroji.

Rozdíl obou přístupů nespočívá pouze v odlišném přístupu k základním proměnným projektového trojimperativu, nýbrž i v mnoha dalších oblastech projektového řízení. Zevrubnější pohled na konkrétní rozdíly obou přístupů poskytuje následující tabulka.

Tabulka 1: Porovnání tradičních a agilních metodik

	Tradiční metodiky	**Agilní metodiky**
Přístup	Prediktivní	Adaptivní
Měření úspěchu	Dodržení plánu	Přidaná hodnota zákazníkovi; spokojenost zákazníka
Velikost projektů	Větší projekty	Menší projekty
Velikost týmu	Velký tým	Malý/kreativní tým, který je schopen řídit sám sebe
Kvalifikace lidí	Standardní jedinci schopni vyhovět předepsaným procesům, očekávaná fluktuace lidí	Důraz na měkké dovednosti, znalosti a silné stránky jednotlivce
Získání požadavků	Detailní specifikace požadavků na počátku projektu.	Požadavky jsou získávány postupně během dílčích iterací
Postoj k změnám	Snaha o jejich minimalizaci, podléhají řízení změn	Jsou vítané, navyšují zákazníkovu přidanou hodnotu
Náklady na změny	Vysoké	Nízké

Kultura řízení	Direktivní řízení a kontroly	Vůdcovství a spolupráce
Zapojení zákazníka	Nízké	Vysoké, zákazník je součástí vývoje i samotného řízení
Vztah zákazníka a projektového týmu	Na základě smlouvy či domluvy	Vzájemná důvěra a spolupráce
Dokumentace	Detailní a rozsáhlá	Snaha o její minimalizaci, důraz především na samotný produkt
Komunikace	Písemná – dokumentace, logy	Osobní kontakt, tváří v tvář
Zaměření	Na procesy	Na lidi
Předběžné plánování	Obsáhlé	Minimální
Návratnost investice	Po ukončení projektu	Inkrementálně už od ukončení první iterace

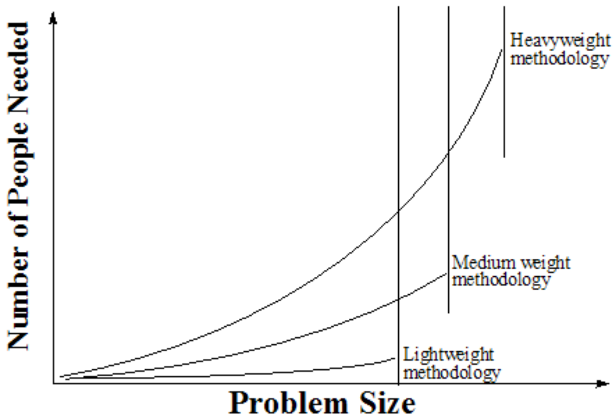
Zdroj: vlastní tvorba dle [19, s. 64] [43] [44] [45]

Z tabulky je na první pohled jasné, že tradiční a agilní přístup toho mají společného pramálo, nicméně otázkou je, zdali je vůbec vhodné tyto dva přístupy takto srovnávat. Na oba přístupy se proto ještě podíváme s ohledem na různé projektové faktory – velikost projektu, lidský faktor, riziko projektu.

6.1 Srovnání z hlediska velikosti projektu

Ještě předtím, nežli přejdeme k srovnání z hlediska velikosti projektu, bylo by vhodné stručně vysvětlit kritérium váhy metodiky. U jednotlivých metodik lze stanovit jakési kritérium váhy, které bere v potaz jejich podrobnost, přesnost, relevanci, toleranci a měřítko, a na jehož základě pak lze dělit metodiky na lehké, za které se zpravidla považují metodiky agilní, a metodiky těžké, za které se považují metodiky tradiční [19, s. 23-24].

Vezmeme-li v potaz velikost projektu, vyjdeme na základě tabulky z předpokladu, že agilní metodiky jsou určené pro malé projekty, zatímco tradiční metodiky lze spíše využít pro větší projekty. Velikostí projektu se pak má na mysli délka projektu, rozpočet a především počet zúčastněných lidí. Alistair Cockburn, spoluautor Agilního manifestu, vytvořil schéma, viz Obrázek 11, kde popisuje vztah velikosti problému a počtu lidí potřebných k jeho vyřešení s ohledem na váhu metodiky [46]. Na obrázku je vidět, že těžší metodiky vyžadují více lidí pro vyřešení problému o dané velikosti v porovnání s metodikami lehčími. Dále je vidět, že lehčí metodiky jsou z hlediska velikosti problému dostačující jen do určité úrovně. Je-li problém větší, nežli umožňuje limitace metodiky, pak je nezbytné přejít na metodiku těžší.



Obrázek 11: Vztah velikosti problému, potřebných lidí a váhy metodiky
Zdroj: **[46]**

Z obrázku lze vyvodit, že určité problémy, které by vyžadovaly velký tým s uplatněním těžké metodiky, lze řešit i za pomocí menšího týmu s uplatněním lehčí metodiky. V takovém případě poskytují agilní metodiky vyšší produktivitu členů projektového týmu a umožňují pružnější řešení daného problému, zatímco tradiční metodiky jsou zbytečně těžkopádné [47]. Nicméně dosáhne-li problém určité limitní velikosti, důsledkem čehož vyžaduje k řešení větší množství lidí, pak lehké metodiky již neposkytují dostatečný rámec a nástroje k efektivnímu řešení onoho problému, a je tedy na místě zvolit metodiku těžkou, zpravidla tradiční. Právě tradiční metodiky totiž umožňují s pomocí pevně stanovených procesů, detailní dokumentace a direktivního řízení lepší koordinaci a komunikaci velkého počtu lidí [46]. Řešením z hlediska agilních metodik pak může být dekompozice problému na dílčí menší problémy, nebo dekompozice velkého projektového týmu na menší sub-týmy [44].

6.2 Srovnání z hlediska lidského faktoru

Lidský faktor hraje při výběru metodiky zcela klíčovou roli, přičemž obě metodiky se k němu staví velmi odlišně. V případě tradičních metodik lze říci, že lidský faktor je sekundární, přičemž primární jsou především kvalitní procesy, které mají zaručovat výslednou kvalitu produkovaného řešení. Potenciál lidského faktoru je tak částečně utlačován na úkor pevných procesů a lidé jsou stavěni do pozice, kdy jsou lehce nahraditelní [19, s. 64]. Naopak agilní metodiky považují členy projektového týmu, jejich kvality a lidský potenciál obecně za primární faktor úspěchu. Při bližším pohledu na Agilní manifest softwarového vývoje zjistíme, že půlka manifestu se věnuje právě lidskému faktoru, kdy pojednává o důležitosti kooperace, jednotlivců a spolupráci se zákazníkem [21]. V případě agilních týmů je kladen velký důraz na rozvoj lidského potenciálu, týmy jsou tvořeny zkušenými a schopnými jedinci, kteří jsou pro tým klíčoví a těžce nahraditelní. Z hlediska lidského faktoru nelze opomenout i účast zákazníka na projektu, která je u tradičních metodik minimální, zatímco agilně řízené projekty považují participaci zákazníka za důležitou součást budoucího úspěchu.

Další důležitou součástí této problematiky je podniková kultura nebo národní mentalita. Má-li podnik dlouhodobou tradici a pevně zavedené podnikové procesy, které nevycházejí z agilních předpokladů, pak může být agilní řízení projektů v rámci takového podniku velmi komplikované a zbytečně odporuje zavedené tradici. K úspěšné integraci agilního přístupu je třeba, aby podnik i zúčastnění lidé přijali agilní filosofii, která považuje změny za přirozenou součást projektu a která vychází z dynamiky dnešního podnikatelského prostředí [22, s. 13]. Obdobnou roli může hrát v uplatnění konkrétních metodik i národní mentalita, viz kapitola Hodnocení Extrémního programování, která pojednává o odlišné aplikovatelnosti XP v podmínkách České republiky a USA.

6.3 Srovnání z hlediska rizika plynoucího z projektu

Vliv na konkrétní formu projektového řízení má i závažnost projektu. Je-li projekt pro organizaci kritický a na jeho úspěchu, resp. neúspěchu závisí budoucnost organizace, pak lze předpokládat, že zadavatel bude preferovat jistoty ve formě detailního plánu, jaké nabízí tradiční přístup, před cestou do neznáma v podobě agilního přístupu [44, s. 37]. Tradiční metodiky tyto jistoty zajišťují právě vysokou úrovní formality v podobě podrobné dokumentace, definovaných procesů a tím, že kladou velký důraz na specifikaci požadavků a pečlivé plánování [19, s. 62]. Je-li primárním cílem agilního přístupu rychlá tvorba přidané hodnoty zákazníkovi, pak lze u tradičního přístupu za primární cíl považovat vysoké záruky a jistoty, které dodržením plánu tento přístup sleduje [47].

V případě, že je v zájmu zadavatele z nějakého důvodu použít agilní metodiky na projekty, které jsou z hlediska organizace kritické, nebo naopak použít tradiční přístup u méně závažných projektů, pak může být řešením oba způsoby nějakým způsobem kombinovat. Kombinací obou přístupů lze dosáhnout odlehčení a zvýšení flexibility u tradičních metodik nebo zvýšení formality a snížení celkového rizika u agilních metodik [19, s. 62]. Obdobně je na základě závažnosti projektu určena úroveň formálních a bezpečnostních opatření u metodik Crystal [42].

7 Případová studie

Zbytek této kvalifikační práce bude věnován případové studii, ve které si stručně přiblížíme reálné projektového prostředí spolu s ilustračními projekty, jež byly v minulosti úspěšně implementovány s využitím vlastní firemní metodiky. Náplní této kapitoly pak bude především návrh alternativy k řízení daných projektů za použití některé z dříve zmíněných agilních metodik.

Případová studie přímo vychází z praktické části bakalářské kvalifikační práce s názvem Uplatnění ICT podpory v systému evidence studentů [48], kde jsou vybrané projekty popsány z hlediska jejich technické implementace s využitím konkrétní softwarové platformy. Tentokrát však bude na oba projekty nahlíženo především s ohledem na problematiku projektového řízení, nikoliv na jejich technické řešení.

7.1 Stručné představení projektového prostředí

Ilustrační projekty byly uskutečněny v rámci holdingu, který zastřešuje několik firem, jež dohromady čítají něco přes 1 100 zaměstnanců. Součástí tohoto holdingu jsou také dvě společnosti, které jsou z hlediska této případové studie stěžejní. Jedná se o soukromou vysokou školu, která vystupuje v roli zadavatele ilustračních projektů, a softwarovou společnost, zabývající se vývojem a správou webových aplikací, která je řešitelem předmětných projektů a v rámci které byly tyto projekty řízeny.

Je důležité podotknout, že vybrané projekty byly vypracovány uvnitř jednoho holdingu, kde platí jednotná firemní kultura, která prosazuje neformální a uvolněnou pracovní atmosféru. Veškerá komunikace napříč holdingem, tzn. i projektová komunikace mezi zadavatelem a řešitelem ilustračních projektů, tedy probíhala formou tykání a v neformálním duchu.

7.1.1 Projektový tým

Ilustrační projekty byly implementovány za pomoci malého projektového týmu, který byl součástí větší firemní jednotky, jež měla za úkol interní vývoj aplikací v rámci holdingu. Celou jednotku tvořilo zhruba 15 členů, kteří však byli rozděleni do samostatných dílčích projektových týmů. Výsledný projektový tým podílející se na realizaci projektu čítal obvykle 4-5 lidí, přičemž každý z nich zastupoval některou z následujících rolí:

- **Projektový manažer** – v celé jednotce byl pouze jeden projektový manažer, který řídil všechny dílčí projektové týmy a který měl přímou zodpovědnost za úspěšnou realizaci všech rozpracovaných projektů. Náplní práce tohoto manažera bylo rozhodování v klíčových otázkách, řízení jednotlivých týmů i pracovníků, kontrola odvedené práce, hodnocení vývoje projektů a řešení naskytnutých problémů a komplikací.
- **Konzultant/Designér** – každý tým obsahoval obvykle dva konzultanty, kteří disponovali podrobnou znalostí technické platformy a kteří měli za úkol komunikaci s klientem, návrh konkrétního řešení a tvorbu dokumentace. Tito lidé se následně podíleli na tvorbě daného řešení v takovém rozsahu, který umožňovala znalost dané platformy, a uživatelsky testovali vyprodukovaný software.
- **Vývojář/Developer** – programátor, který na základě technické dokumentace vytvářel a testoval zdrojový kód. Každý tým obsahoval obvykle jednoho vývojáře, jehož náplní práce byla automatizace vybraných procesů, které konzultant předem definoval v technické dokumentaci.

Vzhledem k vytíženosti projektového manažera byla úroveň řízení uvnitř jednotlivých týmů celkem nízká. Jinými slovy pokud nenastal nějaký problém a vývoj projektu odpovídal plánu, pak manažer ponechal řízení projektu na týmu samotném. Organizace práce se pak odvíjela předně od práce konzultantů, kteří tvorbou technické dokumentace, jež obsahovala i detailní zadání jednotlivých skriptů, přímo vytvářeli podněty pro práci vývojářů, kteří na základě dokumentace mohli započít práce na skriptech, resp. zdrojových kódech.

Je třeba dodat, že v rámci jednotky byly vypracovávány projekty malé, na jejichž úspěšnou implementaci stačil vždy tým maximálně o 6 lidech. Velikost příslušných projektů se pak lišila především očekávaným počtem hodin, které měl daný tým pracemi na konkrétním projektu strávit. Časový odhad se pohyboval u opravdu malých projektů kolem pár set hodin, přičemž za větší se v rámci jednotky považovaly projekty, jež přesáhly hranici 1000 hodin.

7.1.2 Proces řízení projektu a vývoje produktu

K řízení projektů se využívala firemní metodika, která částečně vychází z principů RUP. Obdobně jako tomu je u RUP, i tyto projekty byly dekomponovány na jednotlivé případy užití (use-cases), které byly iterativně vyvíjeny. Ve skutečnosti se však nejednalo o iterativní přístup v pravém slova smyslu, jelikož implementované případy užití byly prezentovány vysoké škole najednou formou „velkého třesku“ až na závěr projektu.

Každý projekt byl na počátku zevrubně analyzován a s pomocí modelovacího jazyka UML byly jednotlivé případy užití v rámci návrhu modelovány. Produkovaný software i dokumentace podléhaly přísným standardům, které měly za úkol produkty a formální dokumenty napříč holdingem sjednotit a zajistit určitou úroveň kvality. V neposlední řadě bylo důležitou součástí projektu také řízení změn. Změny byly řízeny jak z hlediska vtahu zákazník-projektový tým, tak i konzultant-vývojář, přičemž šlo především o změny týkající se technické dokumentace a příslušného zdrojového kódu.

Z procesního hlediska byly projekty zahájeny vždy informační schůzkou projektového týmu se zadavatelem, kde byly specifikovány cíl, vize i požadavky projektu. Uskutečnění počáteční informační schůzky odstartovalo fázi návrhu, kdy byly jednotlivé požadavky podrobně prozkoumány, a mohlo se začít pracovat na návrhu vyvíjené aplikace. Jakmile byl návrh dokončen, byl představen klientovi, který jej buď schválil a tím posvětil započetí samotného vývoje produktu, nebo jej s výhradami vrátil řešiteli a celý proces se opakoval.

Od počátku projektu, tj. od počáteční informační schůzky, až po jeho ukončení se pravidelně konaly také interní schůzky, na kterých se scházel projektový manažer s konzultanty. Vývojáři na těchto schůzkách z nějakého důvodu přítomni nebyli. Konkrétně se jedná o následující dva meetingy:

- **Desetiminutovka**, krátká informativní schůzka konaná na denní bázi, která měla za cíl zběžně informovat projektového manažera o aktuálním stavu projektu. Cíle schůzky se v mnohém podobaly těm z Daily Scrum Meetingu, přičemž cílem tedy bylo pojednat o vykonané práci od poslední schůzky a nastínit pracovní náplň na následující pracovní den.
- **Status Assessment**, cca hodinu dlouhá schůze, která byla konána vždy na konci měsíce. Cílem této schůze bylo zevrubné prozkoumání a zhodnocení aktuálního vývoje projektu. V případě nepříznivého vývoje pak byla prodiskutována nutná opatření tak, aby došlo k zvrácení nepříznivého stavu. Oproti Desetiminutovkám

byly na těchto schůzkách vizuálně prezentovány dosažené výsledky z předešlého měsíce. Jednalo se o jakési interní ukončení měsíční iterace.

Samotný vývoj pak v rámci projektového týmu probíhal rádoby iterativně a po jednotlivých use-casech, nicméně dokončená práce byla prezentována pouze projektovému manažerovi, nikoliv samotnému zákazníkovi. Tímto způsobem byla zajištěna kvalita a funkčnost výsledného softwaru, avšak projektový manažer nemohl 100% zaručit, že výsledný software odpovídá aktuálním představám zákazníka, kterému byl produkt prezentován formou „velkého třesku“ až při jeho dokončení. Nehledě na to, že chyběla průběžná zpětná vazba a nedocházelo k uživatelskému testování dílčích aplikačních inkrementů ze strany vysoké školy. Celý projekt byl završen závěrečnou schůzkou, na které byla klientovi prezentována kompletní aplikace a zároveň byly předány formální dokumenty. V případě, že produkt nebyl včas připraven nebo nebyl během schůzky finálního předání akceptován, byla schůze naplánována na základě času nezbytného k dokončení na další termín a vývoj produktu pokračoval.

Shrneme-li to, zapojení zákazníka do projektu probíhalo průběžně pouze v případě nejasností nebo výskytu nějakého problému, nicméně jakákoliv průběžná kontrola vývoje či zpětná vazba neexistovala. Zapojení bylo tedy pouze omezené a ke skutečné spolupráci došlo obvykle až v rámci servisu při ostrém provozu, kdy se vyskytly nějaké dříve nezjištěné problémy. Komunikace s klientem během projektu pak probíhala především formou elektronických zpráv, telefonicky a pouze výjimečně osobně.

7.1.3 Úskalí použitého přístupu

Výsledkem výše uvedeného přístupu byla obvykle velmi obsáhlá dokumentace obsahující velké množství UML diagramů, která sloužila jako hlavní předloha vyvíjeného produktu. Právě detailní dokumentaci lze považovat vzhledem k charakteru projektů a celého projektového prostředí za zbytečné úskalí celého přístupu. Jakákoliv změna provedená během realizace projektu musela být totiž zároveň promítnuta do dokumentace, přičemž úprava v technické specifikaci a UML diagramech často několikanásobně převýšila časovou náročnost, kterou vyžadovala implementace dané změny v rámci samotného softwaru. Nehledě na to, že jakmile se blížil termín dokončení projektu, byl prostor pro úpravy v dokumentaci čím dál menší a vznikala nekonzistence mezi dokumentací a reálným produktem.

Další úskalí lze najít v komunikaci mezi členy týmu i v rámci vztahu klient a projektový tým. Vzhledem k neformální firemní kultuře a skutečnosti, že si všichni tykají, lze považovat absenci osobního kontaktu za nevyužití potenciálu plynoucího z firemního prostředí. V rámci týmu pak často docházelo ke zbytečně komplikované komunikaci skrze komentáře v dokumentaci namísto přímého a osobního jednání.

Důležité je zmínit také absenci jakéhokoliv optimalizačního nástroje, kterým by projektový tým cílil na zvýšení celkové efektivity projektového řízení a optimalizaci zavedených procesů.

7.2 Představení ilustračních projektů

Ilustrační projekty byly vypracovány pro soukromou vysokou školu, jejíž studijní informační systém je postaven na softwarové platformě, která je využívána k tvorbě interních softwarových řešení v celém holdingu. Z tohoto hlediska byly projekty unikátní, jelikož strana zadavatele měla povědomí o použité technologii a z vlastní iniciativy poskytla na počátku nejen seznam požadavků, ale i hrubý návrh konkrétního softwarového řešení s využitím dané platformy. Cílem projektů pak bylo poskytnutý návrh analyzovat, optimalizovat a nakonec jej implementovat.

Stručné specifikace vybraných projektů, které byly realizovány v rámci spolupráce se soukromou vysokou školou, obsahuje následující text.

7.2.1 Charakteristika projektu A

Cílem prvního projektu byla automatizace procesu uznávání předmětů, které studenti vystudovali na jiné vysoké škole. Automatizací daného procesu mělo dojít k nemalé finanční úspoře vlivem zrychlení celého procesu a snížením administrativních úkonů, které musely být ručně vykonávány zaměstnanci vysoké školy.

Na projektu se podílel čtyřčlenný projektový tým, který sestával z projektového manažera, dvou konzultantů a jednoho vývojáře. Časová náročnost projektu se s uplatněním zvolené metodiky pohybovala okolo 400 hodin.

7.2.2 Charakteristika projektu B

Cílem druhého projektu byla optimalizace procesů týkajících se správy zkouškových termínů, přihlašování studentů na zkoušky a následný zápis známek do elektronických indexů. Implementací této aplikace bylo opět dosaženo značné automatizace a přidání nových funkcí do původního studijního informačního systému, který daná vysoká škola využívala. Tento projekt byl z hlediska fungování vysoké školy kritický, jelikož správa zkouškových termínů a z nich plynoucích výsledků jednotlivých studentů tvoří nezbytnou součást fungování vysokých škol.

Na projektu se podílel totožný projektový tým jako u Projektu A, nicméně větší komplexita projektu vedla k vyšší časové náročnosti, která byla v tomto případě zhruba trojnásobná oproti prvnímu projektu, tzn. cca 1200 hodin.

7.3 Návrh agilního řešení projektů s uplatněním metodiky Scrum

Vzhledem k charakteru projektového prostředí, podnikové kultuře napříč celým holdingem a zavedeným procesům, lze považovat Scrum za velmi vhodnou metodiku, jelikož spousta používaných nástrojů a meetingů velmi blízce korespondují s nástroji, které metodika Scrum nabízí. Přihlédneme-li ke skutečnosti, že Scrum neobsahuje žádný detailní předpis, jak předmětný projektový produkt vyvíjet, a tuto část projektu považuje za empirickou, můžeme s ohledem na projektový tým zachovat původní vývojové procesy a vyhnout se významnému reengineeringu těchto procesů plynoucímu ze zavedení nové metodiky, což lze z hlediska integrace nové metodiky považovat za velké ulehčení.

V následujícím textu si podrobněji popíšeme, jak by mohla vypadat organizační struktura a procesní hledisko výše zmíněných projektů v případě aplikace této agilní metodiky.

7.3.1 Organizační struktura

Scrum rozlišuje dvě kategorie lidí, kterými jsou Pigs a Chickens, kdy Pigs jsou lidé, jež se aktivně podílejí na projektovém řízení a vývoji. Za Chickens pak lze považovat všechny ostatní, v jejichž zájmu je projekt vyvíjen a jichž se projekt týká. Z hlediska organizační struktury jsou pro nás důležití především Pigs, u nichž Scrum definuje speciální projektové role – Product Owner, Scrum Master a Scrum Developer.

Zavedením role Product Ownera do projektového řízení se sleduje navýšení spolupráce a zapojení vysoké školy do projektu. Nedostatečná komunikace byla jedním z hlavních úskalí původního přístupu, což by neměl být nadále problém, jelikož Product Owner zaštítí komunikaci projektového týmu s vysokou školou, a Scrum Master zajistí, aby proaktivní komunikace probíhala i uvnitř projektového týmu.

Product Owner je z hlediska organizační struktury nová role, která nebyla, nebo pouze jen částečně existovala v původně použitém přístupu. Tuto roli může nejlépe zastupovat některá kompetentní osoba ze strany soukromé vysoké školy. V našem případě by se jednalo nejspíše o správce využívaného studijního informačního systému, který se osobně podílel na tvorbě počátečního hrubého návrhu aplikací. Tento člověk je v přímém kontaktu s vedením vysoké školy, které oba projekty iniciovalo a které disponuje jasnou vizí, jež by měly oba projekty naplnit. Další výhodou je, že tato osoba disponuje dostatečnými znalostmi technického i procesního hlediska dané vysoké školy, což jsou předpoklady, které mohou bohatě přispět k efektivnímu plnění práce role Product Ownera. Nicméně nehledě na tento návrh by konečné jmenování Product Ownera spočívalo nejspíše na zadavateli, tzn. na dané vysoké škole.

Náplň práce Product Ownera je:

- tvorba a správa Product Backlogu,
- určování priorit jednotlivých položek Product Backlogu,
- efektivní tlumočení požadavků vysoké školy projektovému týmu,
- zprostředkování komunikace mezi vysokou školou a projektovým týmem,
- kontrola vykonané práce,
- rozhodování v klíčových otázkách týkající se podoby finálního produktu.

Scrum Master je v navrhovaném řešení role, která nahradí pozici tradičního projektového manažera. Definováním role Product Ownera jsme částečně pokryli náplň práce původního projektového manažera, zbytek povinností pak připadne právě roli Scrum Mastera. Tento člověk bude účinkovat především v roli kouče a týmového mentora, je-li třeba, poskytne radu nebo tým usměrní, v opačném případě nechá tým, aby řídil sám sebe. Jeho úkolem je zajistit, aby byl projekt řízen a vývoj probíhal s ohledem na klíčové praktiky metodiky Scrum.

Náplní práce Scrum Mastera je:

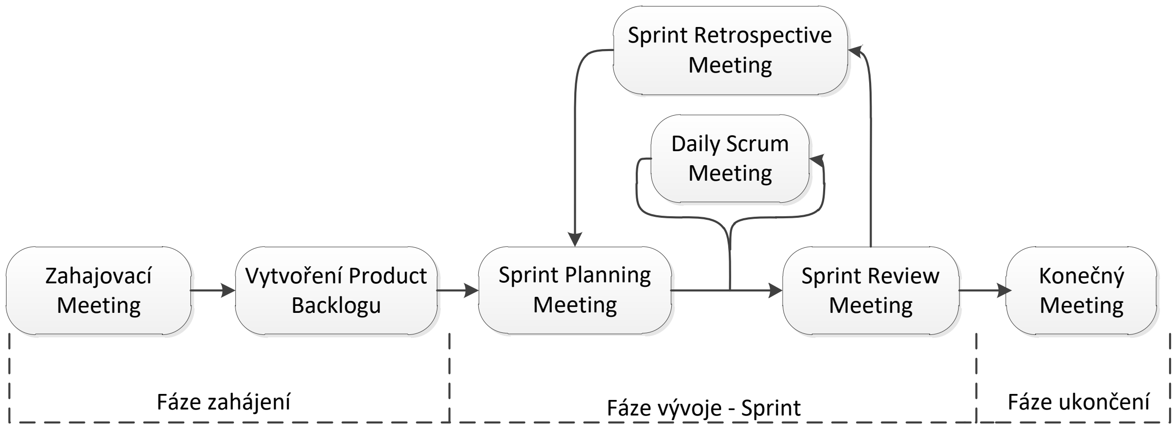
- pomoci týmu dosáhnout vytyčených cílů,
- odstranění či řešení naskytnutých problémů,
- pomoci týmu rozhodnout v technických otázkách,
- motivování týmu,
- moderování projektových schůzek,
- ochrana týmu před vnějšími vlivy, které by mohly tým rozptylovat a odvádět jej od vykonávané práce.

Vzhledem k přenesení části pracovní náplně na Product Ownera lze zachovat původní koncept, kdy jeden projektový manažer organizoval práci několika projektových týmů. Organizační struktura firemní jednotky by se z tohoto hlediska neměnila, pouze by každému projektu navíc náležel konkrétní Product Owner, který by zajistil, aby daný projekt byl vyvíjen správně a s ohledem na aktuální požadavky zákazníka. Původní roli projektového manažera by pak nahradil právě Scrum Master.

Scrum Developer je jednotné označení pro všechny ostatní členy projektového týmu, jež se podílí na tvorbě přidané hodnoty. V tomto konkrétním případě by tato role zahrnovala obě původní role – konzultanta i vývojáře. Obě role by pokračovaly ve své dosavadní činnosti, jen by byl kladen větší důraz na vzájemnou spolupráci a aktivní komunikaci. Optimálně by měl projektový tým řídit sám sebe, což je dlouhodobým cílem Scrum Mastera.

7.3.2 Procesní hledisko

Životní cyklus projektu řízeného metodikou Scrum lze rozdělit na tři fáze, viz Obrázek 12. První fází je fáze zahájení (Předehra), následuje iterační fáze (Hra), během které dochází k jednotlivým Sprintům, a celý projekt je završen fází ukončení (Dohra). Nedílnou součástí každé fáze je pořádání specifických setkání, která si popíšeme v následujícím textu, jenž se jednotlivým fázím podrobněji věnuje.



Obrázek 12: Navrhovaný životní cyklus projektu dle metodiky Scrum
Zdroj: vlastní tvorba dle [27]

I. Fáze zahájení (Předehra)

Fáze zahájení je započata počáteční informační schůzí, na které se poprvé sejde projektový tým (Pigs) se zainteresovanými osobami (Chickens). Během této schůzky je představen primární koncept projektu, jeho vize a především je určena role Product Ownera ze strany vysoké školy. V našem případě vycházíme z předpokladu, že projektový tým je stálý, jinak by byli představeni a vzájemně seznámeni i členové nově formovaného projektového týmu.

Po zahajovací schůzce jsou započaty práce na tvorbě Product Backlogu, což je klíčový dokument, resp. seznam, který obsahuje kompletní specifikaci vyvíjené aplikace. Jednotlivé položky Product Backlogu lze formulovat různě, v tomto případě, kdy je vyvíjen software, lze použít uživatelské příběhy (User Story), které reprezentují jednotlivé funkcionality softwaru. Nicméně lze ponechat i původní případy užití (Use Case), které se uživatelským příběhům v mnohém podobají.

Product Backlog je obvykle strukturován do tabulky, která obsahuje jednotlivé uživatelské příběhy a další důležité informace jako jsou priorita příběhů a stav. Dále tento dokument může obsahovat náročnost implementace nebo položky kategorizovat. V případě projektu B by mohla část Product Backlogu vypadat třeba následovně:

Tabulka 2: Ukázka artefaktu - Product Backlog

ID	Uživatelský příběh	Priorita	Stav
1	Pedagog vypíše nový zkouškový termín	Vysoká	Hotovo
2	Pedagog upraví vlastnosti zkouškového termínu	Střední	
3	Student se přihlásí na zkouškový termín	Vysoká	Probíhají práce
4	Student se odhlásí ze zkouškového termínu	Vysoká	Hotovo
5	Pedagog odhlásí studenta ze zkouškového termínu	Nízká	
6	Pedagog zpracuje výsledky zkouškového termínu	Střední	
7	Pedagog zruší zkouškový termín	Střední	
…	…	…	…

Zdroj: vlastní tvorba

Na tvorbě Product Backlogu se podílejí různí lidé, kteří mohou zastupovat zákazníka, náležet do projektového týmu nebo jsou experty v dané problematice. Ve fázi zahájení je pak vytvořen základní koncept Backlogu, přičemž je snaha obsáhnout co nejvíce požadavků v co nejhlubší míře. Nicméně je třeba počítat s tím, že celý dokument se bude měnit a vyvíjet během projektu s ohledem na aktuální požadavky ze strany vysoké školy.

II. Fáze vývoje (Hra)

Vytvořením Product Backlogu je ukončena fáze zahájení, a je možné započnout fázi vývoje. Scrum považuje fázi vývoje za empirickou část řízení projektu, přičemž konkrétní podobu vývojových procesů a používaných nástrojů ponechává na rozhodnutí projektového týmu. Považuje-li tedy projektový tým původní vývojové procesy za dostatečně efektivní, pak lze tyto procesy ponechat v rámci nového přístupu z velké části zachovány.

Celá fáze vývoje je nicméně členěna do jednotlivých iterací nazývaných Sprinty, přičemž každý Sprint vychází z procesního hlediska, které definuje metodika Scrum. Procesním hlediskem tentokrát nemáme na mysli samotný vývojový proces, nýbrž konkrétní nástroje projektového řízení. Jedná se především o formální schůzky, jež nahradí dříve využívané Desetiminutovky a Status Assessment. Všechny tyto schůzky jsou přehledně vyobrazeny na Obrázku 12.

V pořadí první formální schůze se nazývá **Sprint Planning Meeting** a jejím účelem je vytvoření Sprint Backlogu. Na jeho vytvoření se podílí celý projektový tým, který na základě priorit jednotlivých položek Product Backlogu vybere ty, jež budou implementovány v následující iteraci. Sprint Backlog bude mít opět formu strukturované tabulky, která obsahuje položky implementované v aktuálním Sprintu. Názorná ukázka části Sprint Backlogu je k vidění v Tabulce 3. Počet položek ve Sprint Backlogu se bude lišit s ohledem na náročnost implementace jednotlivých položek a délku Sprintu. V této případové studii se jedná o projekty malé s relativně stabilními požadavky, proto lze nastavit iterace o délce např. třech týdnů, čímž bude docíleno frekventované zpětné vazby ze strany vysoké školy při zachování rozumného celkového počtu iterací. Tři týdny jsou kompromisem mezi dlouhými a krátkými iteracemi, přičemž poskytují dostatek prostoru k tvorbě značné přidané hodnoty pro zákazníka během jednoho cyklu, avšak udržují zákazníka zapojeného do procesu vývoje častými prezentacemi dosažených výsledků.

Tabulka 3: Ukázka artefaktu – Sprint Backlog

					Zbývá hodin			
Položka Product Backlogu	**Úkol**	**Kdo**	**Stav**	**Časový odhad**	**Den 1**	**Den 2**	**Den 3**	**Den 4**
Student se přihlásí na zkouškový termín	Návrh logiky	Jan	Hotovo	7	4	0	0	
	Návrh a realizace UI	Jan	Hotovo	15	15	11	3	
	Implementace zdrojového kódu a testování	Ota	V řešení	20	15	8	0	
Pedagog vypíše nový zkouškový termín	Návrh logiky	Jan	Hotovo	10	5	5	5	
	Návrh a realizace UI	Jan	V řešení	20	20	20	20	
	Implementace zdrojového kódu a testování	Ota	V řešení	23	20	19	19	
		Celkově zbývá		**95**	**79**	**63**	**47**	

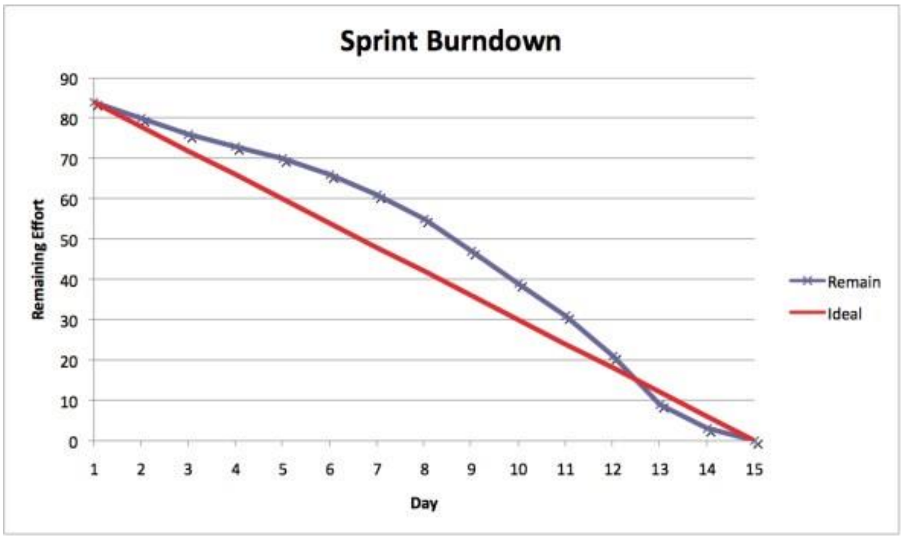
Zdroj: vlastní tvorba

Je-li vytvořen Sprint Backlog, mohou započít práce na vývoji produktového inkrementu, v našem případě se jedná o implementace jednotlivých uživatelských příběhů. Scrum proces vývoje považuje za empirickou část projektu a nechává na projektovém týmu, jaké procesy, nástroje a přístup k vývoji použijí. Lze tedy ponechat původní vývojový proces, avšak je v zájmu obou zúčastněných stran provést nepatrné změny, které eliminují zmíněná úskalí původního přístupu.

Na prvním místě je tvorba dokumentace, která celý proces vývoje zpomalovala a činila jej zbytečně těžkopádným. Dokumentace je důležitá, nicméně výsledný produkt je přece jen přednější – nemělo by tedy docházet k vytváření dokumentace na úkor skutečného vývoje. V původním přístupu mohla její tvorba a následné úpravy pokrýt až třetinu celkového času stráveného na projektu. Z tohoto hlediska by bylo vhodné zavést zásady štíhlého přístupu a z dokumentace eliminovat vše, co není nezbytně nutné a co nepřináší přidanou hodnotu. Zároveň by měla být dokumentace vytvářena v nejzazší možný termín tak, aby bylo možné předejít velmi častým modifikacím, které se negativně promítaly do celkového rozpočtu. Vlivem tohoto zeštíhlení se očekávají nezanedbatelné časové i finanční úspory současně s navýšením agility celého procesu vývoje.

Dalším úskalím byla nedostatečná osobní komunikace v rámci a vně projektového týmu. Komunikace by měla optimálně probíhat osobně a v reálném čase formou aktivní spolupráce, přičemž stranu zákazníka reprezentuje především Product Owner, který je v blízkém kontaktu se zástupci vysoké školy i projektovým týmem. Tento a další problém, kterým je absence jakékoliv formy zdokonalování zavedených procesů, řeší série formálních schůzek, které metodika Scrum zavádí během jednotlivých Sprintů. Náplní Scrum Mastera je pak tyto schůze pořádat a moderovat.

Daily Scrum Meeting je každodenně pořádaná schůzka, jejíž délka by neměla přesáhnout 10 minut. Jedná se o schůzku typově velmi podobnou původně pořádaným Desetiminutovkám, které z procesního hlediska Daily Scrum přímo nahrazuje. Účelem této schůzky je udržovat informovanost týmu o aktuálním stavu vývoje projektu. Za tímto účelem může být použito speciálního nástroje – Burndown diagramu, který je zobrazen na Obrázku 13, kde červená linie zobrazuje optimální projektový vývoj a modrá linie vývoj skutečný. Má-li projekt zpoždění, bude linie aktuálního stavu nad linií optimálního průběhu, a naopak je-li projekt napřed, bude modrá linie aktuálního stavu pod tou červenou, reprezentující ideální případ.



Obrázek 13: Burndown diagram
Zdroj: Sprint Burndown Reports / Charts [online]. [cit. 2016-11-20]. Dostupné z: http://www.scrum-institute.org/images_scrum/Sprint_Burndown.jpg

Daily Scrumu se účastní ideálně všichni Pigs, tzn. celý projektový tým v čele se Scrum Masterem, který schůzi moderuje, a Product Ownerem, který účinkuje v roli pozorovatele, a je-li třeba, poskytuje zpětnou vazbu. Každý Scrum Developer musí během této krátké schůze ještě stručně odpovědět na tři otázky týkající se odvedené práce od posledního setkání, náplni práce před dalším setkáním a výskytu jakéhokoliv problému, který by vyžadoval pozornost ostatních členů týmu.

Sprint Review Meeting je oficiální ukončení Sprintu z hlediska vývoje produktového inkrementu. V našem návrhu by se tato schůze konala jednou za tři týdny, kdy by byly prezentovány výsledky právě ukončeného Sprintu. Na tomto setkání se sejde projektový tým spolu s lidmi reprezentující stranu zákazníka, přičemž schůze je vedena v neformální atmosféře, kdy lidé z vysoké školy mají prostor vyzkoušet vytvořený produktový inkrement, resp. implementované uživatelské příběhy. Účelem projektového týmu v čele se Scrum Masterem je kromě prezentace dosažených výsledků také zodpovězení případně kladených otázek. Tento meeting částečně nahrazuje Status Assessment, který původně sloužil k interní prezentaci dosažených výsledků na konci každého měsíce a kontrole stavu vývoje projektu. Výsledky jsou nově prezentovány průběžně na konci každého Sprintu zákazníkovi a stav je kontrolován denně v rámci Daily Scrum meetingů.

Sprint Retrospective Meeting je posledním formálním setkáním, které náleží aktuálnímu Sprintu a kterým je každý Sprint definitivně uzavřen. Na tomto meetingu se sejde pouze projektový tým, přičemž jeho účelem je kriticky se ohlédnout na uplynulou iteraci a hledat prostor k budoucímu zlepšení. Absence jakékoliv optimalizace a zlepšování zavedených procesů bylo jedním z velkých nedostatků předchozího přístupu a zavedením těchto pravidelných setkání je tento nedostatek řešen. Kritika a snaha o optimalizaci se může týkat prakticky čehokoliv, nicméně primárně by se měl tým soustředit na aplikované nástroje a procesy, nejedná se tedy o prostor k řešení osobních sporů.

III. Fáze ukončení (Dohra)

Je-li poslední plánovaná iterace završena a očekávání zástupců vysoké školy naplněna, může projekt přejít do fáze ukončení, kdy probíhají finální přípravy aplikace k nasazení do ostrého provozu, závěrečné testování a tvorba uživatelské dokumentace. Během této fáze se zároveň koná poslední formální setkání, kterým se celý projekt definitivně uzavře. Jedná se o konečný meeting, na němž se sejde projektový tým se zástupci vysoké školy. Během tohoto setkání dojde k oficiálnímu ukončení celého projektu, kdy je finální produkt akceptován stranou zadavatele a spolu s ním jsou předány všechny příslušné projektové výstupy, jako je dokumentace, akceptační protokoly atd. Součástí meetingu může být i hodnocení průběhu projektu nebo domluva o případné údržbě vyvinutých aplikací či budoucí spolupráci.

7.4 Zhodnocení navrhovaného přístupu

Ačkoliv byl původní přístup k řízení obou ilustračních projektů úspěšný, disponoval množstvím úskalí, která zbytečně snižovala celkovou efektivitu řízení projektů i samotné práce vývojářů a přímo přispívala k zvyšování celkových nákladů. Zavedením metodiky Scrum se cílí na integraci agilních principů a odstranění těchto úskalí. Některé nástroje a procesy z původního přístupu pak byly v rámci nového přístupu zachovány, pouze došlo k jejich formálnímu přejmenování a drobné úpravě tak, aby korespondovaly s novou metodikou. Další nástroje a formální schůzky byly následně s nasazením metodiky Scrum nově zavedeny.

Cílem následujícího textu je shrnout přínosy navrhovaného přístupu a blíže popsat některé předpoklady, které jsou nezbytné pro úspěch tohoto přístupu.

7.4.1 Přínosy navrhovaného přístupu

Nepochybným přínosem plynoucím z aplikace metodiky Scrum je větší participace vysoké školy na projektovém řízení i samotném vývoji. Nově je zavedena role Product Ownera, který reprezentuje stranu klienta a zároveň je součástí projektového týmu. Integrací této role do procesu řízení projektů je dosaženo toho, že produkt je vyvíjen s ohledem na aktuální požadavky vysoké školy a podléhá neustálému dohledu kompetentní osoby. Kromě neustálého dohledu Product Ownera je zapojení vysoké školy vynuceno i skrze iterativní přístup, jenž zavádí krátké třítýdenní Sprinty. Tento přístup nově dekomponuje celý proces vývoje na krátké dílčí fáze, které jsou zakončeny prezentací dosažených výsledků zákazníkovi, tzn. zástupcům vysoké školy. Meetingy na konci každého Sprintu pak umožňují zákazníkovi poskytovat zpětnou vazbu a testovat již vyvinuté inkrementy.

Product Owner částečně řeší také otázku komunikace, jelikož zaštiťuje oboustrannou komunikaci mezi projektovým týmem a zákazníkem. V navrhovaném řešení je kladen velký důraz na efektivní a častou komunikaci napříč celým projektem, jelikož metodika Scrum přímo vyzdvihuje důležitost osobního jednání a frekventované komunikace. Není-li tomu tak, je na Scrum Masterovi, aby zavedl konkrétní opatření, která neefektivní komunikaci vyřeší. Z hlediska komunikace jsou pak klíčové především formální schůzky, které jsou skrze metodiku Scrum integrovány do procesu projektového řízení. Tyto schůzky v případě Daily Scrum Meetingů přímo vycházejí z původních Desetiminutovek, nicméně nově jsou zavedeny i další charakteristické schůzky, které původně součástí projektů vůbec nebyly. Jedná se o Sprint Review Meeting a Sprint Retrospective Meeting. Skrze první zmíněnou schůzku je zajištěna frekventovaná zpětná vazba ze strany vysoké školy nad rámec působení Product Ownera. Během této schůzky dochází k osobnímu kontaktu projektového týmu s kompetentními zástupci vysoké školy, kteří mají každé tři týdny příležitost vidět naživo dosažené výsledky Sprintu a v reálném čase na ně mohou reagovat. Druhým zmíněným meetingem je Sprint Retrospective Meeting, který nově zavádí do procesů projektového řízení určitou formu optimalizace procesů skrze retrospektivu. Tato optimalizace zajišťuje dlouhodobou efektivitu celého procesu projektového řízení, přičemž absenci jakéhokoliv optimalizačního nástroje lze považovat u původního přístupu za velmi zásadní nedostatek.

Co se týče samotného vývoje produktu, nově je zaveden regulérní iterativní vývoj[7], který umožňuje frekventovanou zpětnou vazbu a větší zapojení vysoké školy do vývoje. V rámci návrhu je dále doporučeno aplikovat principy štíhlosti, které eliminují jakékoliv plýtvání a zredukují původní velmi obsáhlou dokumentaci, jejíž tvorba se významně podílela na celkovém podílu odpracovaných hodin. Zredukováním dokumentace a zavedením iterativního vývoje umožňuje navrhovaný přístup lépe reagovat na změny, a tím uspořit finanční prostředky plynoucí z implementace změnových požadavků.

Vyjma členění na jednotlivé Sprinty a celkového zeštíhlení nejsou zavedeny v rámci návrhu žádné konkrétní vývojové metody ani nástroje, jelikož tuto fázi projektového řízení považuje metodika Scrum za empirický proces a jeho konkrétní podobu ponechává na rozhodnutí projektového týmu. Považuje-li tedy projektový tým původní vývojové metody a nástroje za dostatečně efektivní, pak samotný vývojový proces nemusí podléhat reengineeringu vyplývajícímu z integrace nové projektové metodiky.

Z hlediska úspor lze o navrhovaném řešení prohlásit, že časové i finanční úspory plynou ze zvýšení efektivity celého projektového řízení vlivem následujících faktorů, které Scrum do projektového řízení nově přináší. Prvním takovým faktorem je zavedení optimalizace formou retrospektivy, která vede z dlouhodobého hlediska k eliminaci slabých stránek celého procesního hlediska. K navýšení efektivity může přispět i aktivní podpora spolupráce a osobní komunikace mezi členy projektového týmu ze strany Scrum Mastera i samotné metodiky. Scrum dále motivuje jednotlivé členy projektového týmu navozením osobního zájmu a zodpovědnosti na projektu vlivem Daily Scrum Meetingů, kde svoji práci každý člen projektového týmu denně prezentuje před ostatními. Původní Desetiminutovky umožňovali z části totéž, nicméně Desetiminutovek se neúčastnil celý projektový tým, čímž byl motivační efekt zčásti eliminován. Klíčovým faktorem z hlediska úspor je však zavedení štíhlosti do celého procesu projektového řízení, čímž je dosaženo významných časových i finančních úspor vlivem eliminace plýtvání a především pak zredukováním velmi obsáhlé dokumentace.

[7] Původní metodika sice vycházela z RUP, jenž je založen na iterativním vývoji, nicméně v tomto konkrétním případě byl celý koncept špatně pochopen, což mělo za následek absenci výše zmíněných přínosů. Původně se tedy jednalo o iterativní vývoj opravdu jen vzdáleně.

7.4.2 Předpoklady úspěchu navrhovaného přístupu

Úspěšné uplatnění metodiky Scrum vychází z několika velmi důležitých předpokladů, které musejí být naplněny. V opačném případě by se tato metodika stala s největší pravděpodobností pouze prostředkem neúspěchu a celkového chaosu.

Prvním předpokladem je zkušený projektový tým, který dokáže řídit sám sebe. Scrum je krajně nevhodný pro projektové týmy bez předešlých zkušeností vzhledem k volnosti a úrovni nezávislosti, jež do týmu vkládá. V případě nezkušeného týmu by pak takováto metodika způsobila spíše chaos a absenci řídícího prvku, což by mělo za následek přesný opak kýženého efektu, tzn. drastický pokles produktivity.

Druhým předpokladem je osobní nasazení všech členů týmu. Jednotliví členové musí být soběstační, zodpovědní a spolehliví. V případě, že některý člen týmu nespolupracuje či neplní svoje povinnosti, pak může být ohrožena práce celého týmu, což by vedlo k selhání projektu. S tímto předpokladem je i úzce spjata skutečnost, že opuštění jakéhokoliv člena týmu během projektu může vést k vážným komplikacím, jelikož každý člen týmu hraje v rámci projektu klíčovou roli.

Posledním předpokladem je malý projektový tým. Scrum lze jen těžko aplikovat na velké projektové týmy, což je opět spjato se schopností týmu řídit sám sebe. Dalším důvodem preference malého týmu je skutečnost, že frekventované formální meetingy by v přítomnosti velkého počtu lidí zbytečně zatěžovaly celý proces a stávaly se tak méně efektivními. V naší případové studii lze považovat velikost týmu o pěti lidech za optimální.

Závěr

Tato kvalifikační práce si dala za cíl podrobněji prozkoumat oblast projektového řízení se zvláštním zaměřením na agilní metodiky. Práce vychází z předpokladu, že správné porozumění agilnímu projektovému přístupu vyžaduje určitou znalost obecné problematiky stejně jako tradičního přístupu. Tento předpoklad vychází ze skutečnosti, že agilní metodiky začaly být vyvíjeny v reakci na nedostatky tradičního přístupu, které se začaly objevovat s rostoucí dynamikou okolního světa. Zevrubnému popisu agilních metodik tedy předchází obecná část, která stručně popisuje problematiku projektového řízení jako celku a následně přechází v detailnější popis tradičního přístupu, jehož součástí je také představení nejvýznamnějších tradičních metodik.

Celá problematika konkrétních metodik projektového řízení je velmi úzce spjata s oblastí vývoje softwaru. Všechny metodiky uvedené v této diplomové práci vznikly primárně za účelem řízení projektů týkajících se softwarového vývoje, výjimku pak tvoří okolnosti vzniku štíhlé výroby a z ní vycházející metodiky Lean Development. Nicméně práce se snaží prezentovat metodiky, je-li to jen možné, v obecnějším slova smyslu z pohledu jejich aplikace na různé typy projektů, jež se netýkají pouze vývoje softwaru. Takový pohled na tuto konkrétní problematiku lze považovat za přínos této práce, jelikož velmi málo literatury se dnes věnuje agilním metodikám s ohledem na projekty mimo oblast softwarového vývoje. Součástí teoretické rešerše jsou však i některé metodiky, jejichž aplikace se týká výhradně vývoje softwaru, jelikož je lze považovat z hlediska agilních metodik za natolik významné, že jejich absence by učinila kvalifikační práci nekompletní. Hlavní přínos teoretické části pak lze spatřit v poskytnutí zevrubného představení problematiky různých přístupů k projektovému řízení včetně konkrétních metodik a závěrečného srovnání agilního a tradičního přístupu.

S ohledem na teoretickou část a především závěrečné srovnání obou přístupů pak lze říci, že tradiční i agilní přístup reprezentují extrémy, které ne vždy správně vystihují reálnou situaci. Velmi často tedy dochází k prolínání obou přístupů, což se konkrétně projevuje i na samotných metodikách, kdy třeba agilní DSDM zavádí určitou úroveň předběžného plánování nebo RUP integruje agilní myšlenky. Oba přístupy pak mají svou komfortní zónu, v rámci které je lze bez váhání aplikovat na daný projekt, a nikdo nebude pochybovat o správnosti daného rozhodnutí. Nicméně čím více je onen konkrétní projekt vzdálen od této komfortní zóny, tím méně bude dedikovaná metodika reprezentující daný přístup vhodná. V takovém případě je vhodné s ohledem na typ projektu skloubit to dobré z obou přístupů a tvrdohlavě nenásledovat metodické předpisy. Závěr lze pak formulovat tak, že i přes narůstající popularitu agilního přístupu má tradiční přístup své nenahraditelné místo ve světě projektového řízení a není-li projekt v komfortní zóně, pak je nejvyšší efektivity dosaženo kombinací potřebných nástrojů a procesů z obou přístupů, jež jsou uplatněny ve vzájemné synergii.

V praktické části se tato kvalifikační práce věnovala případové studii projektového řízení v reálné organizaci, kdy cílem práce bylo prozkoumat použitou metodiku a na základě nabytých poznatků navrhnout agilní alternativu. Vzhledem k charakteru ilustračních projektů i projektového prostředí byla nakonec zvolena metodika Scrum, která perfektně vyhovuje povaze původního organizačního i procesního hlediska a nabízí řešení nedostatků, kterými původní přístup oplýval. Majoritní část případové studie tedy zahrnuje podrobný popis organizační struktury a procesního hlediska ilustračních projektů v případě aplikace metodiky Scrum. Přínos navrhovaného přístupu pak lze nalézt v následujících bodech:

- aktivní zapojení strany zákazníka do projektového řízení,
- efektivní a častá projektová komunikace v projektovém týmu i se zákazníkem,
- zavedení iterativního vývoje a z něj plynoucí frekventovaná zpětná vazba ze strany zákazníka,
- integrace procesního optimalizačního nástroje formou retrospektivy na konci každé iterace,
- zvýšení agility a snížení nákladů vlivem integrace štíhlého přístupu do vývoje.

Výše zmíněné body současně řeší nedostatky původního přístupu a přímo přispívají k navýšení celkové efektivity projektového řízení ilustračních projektů, výsledkem čehož lze očekávat nemalé časové i finanční úspory. Navrhované řešení navíc umožňuje zachování velké části původního vývojového procesu, jelikož Scrum vývojovou fázi považuje za empirickou část řízení projektu a její podobu ponechává na projektovém týmu. Považuje-li tedy projektový tým původní vývojové procesy za dostatečně efektivní, pak je v rámci návrhu eliminována potřeba významnějšího reengineeringu těchto procesů, jenž obvykle ze zavedení nové agilní metodiky řízení projektů vyplývá.

Seznam použité literatury

Citace

[1] What is Project Management?. *PMI* [online]. USA: PMI, 2016 [cit. 2016-11-10]. Dostupné z: https://www.pmi.org/about/learn-about-pmi/what-is-project-management

[2] ISO 10006: Management jakosti - Směrnice jakosti v managementu projektu. Český normalizační institut, 1999.

[3] Management Mania: Řízení projektů (*Project* Management) [online]. MANAGEMENTMANIA.COM LLC, c2011-2016 [cit. 2016-11-10]. Dostupné z: https://managementmania.com/cs/metody-rizeni-projektu

[4] Project Management Institute: Learn About PMI. PMI [online]. USA: PMI, 2016 [cit. 2016-11-10]. Dostupné z: http://www.pmi.org/about/learn-about-pmi

[5] Vision and mission. IPMA [*online*]. IPMA, 2015 [cit. 2016-11-10]. Dostupné z: http://www.ipma.world/about/vision/

[6] WYSOCKI, Robert, Sarah KAIKINI *a* Ryan SNEED. Effective project management: traditional, agile, extreme. Seventh edition. Indianapolis, Indiana: Wiley, 2014. ISBN 978-1-118-72931-1.

[7] 3T Projects: Strategic Project Portfolio Roadmap [online]. 2012 [cit. 2016-11-10]. Dostupné z: http://www.3tprojects.com/Strategic%20Project%20Portfolio%20Roadmap.html

[8] Projectmanagementforum's Blog: The Five Project Management Life Cycles [online]. Thomas Franchina, 2010 [cit. 2016-11-10]. Dostupné z: https://projectmanagementforum.wordpress.com/2010/07/06/the-five-project-management-life-cycles/

[9] HASS, Kathleen The Blending of Traditional and Agile Project Management. Project Management World Today [online]. 2007, 2007(), 1-8 [cit. 2016-09-26]. Dostupné z:

http://chelsoftusa.com/uploads/3/4/1/3/34136265/agile_well_explained.pdf

[10] KADLEC, Václav. Agilní programování: metodiky efektivního vývoje softwaru. Vyd. 1. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

[11] . HUGHES, Bob a Mike COTTERELL. Software project management. 5th ed. London: McGraw-Hill Higher Education, 2009, s. 82-85. ISBN 0077122798.

[12] . CADLE, James a Donald YEATES. Project management for information systems. 4th ed. Harlow: Financial Times Prentice Hall, 2004, s. 79. ISBN 0273685805.

[13] Comparing Traditional Systems Analysis *and* Design with Agile Methodologies: The Traditional Waterfall Approach [online]. University of Missouri-St. Louis: Douglas Hughey, 2009 [cit. 2016-11-10]. Dostupné z: http://www.umsl.edu/~hugheyd/is6840/waterfall.html

[14] JONASSON, Hans. Determining *project* requirements. Boca Raton: Auerbach Publications, 2008. ISBN 14-200-4502-4.

[15] IBM: IBM Professional Certification Program [online]. b.r. [cit. 2016-11-10]. Dostupné z: http://www-03.ibm.com/certify/certs/38008003.shtml#

[16] Introduction to OpenUP (Open Unified Process) [online]. 2007 [cit. 2016-11-10]. Dostupné z: https://eclipse.org/epf/general/OpenUP.pdf

[17] Usability.gov: Use Cases [online]. Washington, D.C, 2016 [cit. 2016-11-10]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/use-cases.html

[18] RATIONAL SOFTWARE, . Rational Unified Process: Best Practices for Software Development Teams [online]. In: . 1998, s. 21 [cit. 2016-11-10].

[19] BUCHALCEVOVÁ, Alena. Metodiky *vývoje* a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. 1. vyd. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1075-7.

[20] AMBLER, Scott W. *A* Manager's Introduction to The Rational Unified Process (RUP). 2005. Dostupné také z: http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

[21] Manifest Agilního vývoje software *[*online]. 2001 [cit. 2016-11-10]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

[22] ŠOCHOVÁ, Zuzana a Eduard KUNCE. Agilní metody řízení projektů. 1. vyd. Brno: Computer Press, 2014, 175 s. ISBN 978-80-251-4194-6.

[23] CERVONE, H. Frank. Understanding agile project management methods using Scrum. OCLC Systems & Services: International digital library perspectives. 2011, , 18-22.

[24] MAC IVER, Robbie. *Scrum* is not just for software: A real-life application of Scrum outside IT. Scrum Alliance. 2009.

[25] Jim Highsmith: What is Agility? [online]. 2010 [cit. 2016-11-10]. Dostupné z: http://jimhighsmith.com/what-is-agility/

[26] Scrum Alliance: Who uses Scrum and why? [online]. Scrum Alliance, 2016 [cit. 2016-11-14]. Dostupné z: https://www.scrumalliance.org/why-scrum/who-uses-scrum

[27] MYSLÍN, Josef. Scrum: průvodce agilním vývojem softwaru. 1. vydání. Brno: Computer Press, 2016. ISBN 978-80-251-4650-7.

[28] Scrum Guide. Scrum Alliance [online]. Scrum Alliance, 2016 [cit. 2016-11-14]. Dostupné z: https://www.scrumalliance.org/why-scrum/scrum-guide

[29] BALLARD, Glenn a Gregory A. HOWELL. Lean project management. 2003, , 1-15.

[30] AZHARUL, Karim a Saviz NEKOUFAR. Lean Project Management. Faculty of Built Environment and Engineering, *Queensland University of Technology*. 2011.

[31] BALLÉ, Freddy a Michael BALLÉ. Lean Development. Business Strategy Review.

2005, , 17-22.

[32] ABRAHAMSSON, Pekka, Nilay *OZA* a Christof EBERT. Lean software development. IEEE Software. 2012, 2012, 22-25.

[33] POPPENDIECK, Mary a Tom POPPENDIECK. Lean software development: an agile toolkit. 10. print. Boston: Addison-Wesley, 2006. ISBN 03-211-5078-3.

[34] Agile Business Consortium [online*]*. 2016 [cit. 2016-11-14]. Dostupné z: https://www.agilebusiness.org/

[35] The DSDM Agile Project *Framework* (2014 Onwards) [online]. DSDM Consortium, 2014 [cit. 2016-10-25]. Dostupné z: https://www.agilebusiness.org/content/introduction-0

[36] DSDM is Agile's Best *Kept* Secret?. Agile Scout: Agile software development news [online]. 2011 [cit. 2016-11-14]. Dostupné z: http://agilescout.com/dsdm-is-agiles-best-kept-secret/

[37] DSDM Atern Project Structure – Overview. M.C. Partners & Associates. [online]. 2011 [cit. 2016-11-14]. Dostupné z: http://www.mcpa.biz/2011/10/dsdm-atern-project-structure-overview/

[38] Extreme Programming [online]. 2009 [cit. 2016-11-14]. Dostupné z: http://www.extremeprogramming.org/

[39] PALMER, Stephen R. *a* John M. FELSING A practical guide to feature-driven development. Upper Saddle River, NJ: Prentice Hall PTR, 2002. ISBN 01-306-7615-2.

[40] BECK, Kent. Test-driven *development*: by example. Boston: Addison-Wesley, 2003. ISBN 03-211-4653-0.

[41] Crystal. John Pruitt: *Perspectives* on Software Development [online]. 2011 [cit. 2016-11-14]. Dostupné z: https://blog.jgpruitt.com/2011/02/12/crystal/

[42] COCKBURN, Alistair. Crystal *clear*: a human-powered methodology for small teams. Boston: Addison-Wesley, 2005. ISBN 02-016-9947-8.

[43] LEAU, Yu Beng, Wooi Khong LOO, Wai Yip THAM a Soo Fun TAN. Software Development Life Cycle AGILE vs Traditional Approaches. *2012 International Conference on Information and Network Technology*. 2012, (37), 162-167.

[44] AWAD, M.A. *Comparison* between Agile and Traditional Software Development Methodologies The University of Western Australia, 2005, , 35.

[45] ŠPUNDAK, Mario. Mixed *agile*/traditional project management methodology – reality or illusion?. Procedia - Social and Behavioral Sciences. 27th IPMA World Congress, 2014, , 939–948

[46] Methodology per project. Alistair Cockburn [online]. 1999 [cit. 2016-11-14]. Dostupné z: http://alistair.cockburn.us/Methodology+per+project

[47] BOEHM, B. *Get ready for agile methods, with care*. Computer. b.r., 35(1), 64-69. DOI: 10.1109/2.976920. ISSN 00189162. Dostupné také z: http://ieeexplore.ieee.org/document/976920/

[48] HALAMA, Petr. *Uplatnění ICT podpory v systému evidence studentů*. Liberec, 2014. Bakalářská práce. Technická univerzita v Liberci. Vedoucí práce Doc. Ing. Klára Antlová, Ph.D.

Bibliografie

[49] HAJDIN, Tomáš. Agilní metodiky vývoje software. Brno, 2005. Diplomová práce. Masar1ykova univerzita v Brně.

[50] HONSOVÁ, Veronika. Projektové řízení v konkrétní společnosti - agilní vs. tradiční přístupy. Praha, 2013. Diplomová práce. Vysoká škola ekonomická v Praze.

[51] PROCHÁZKA, Jaroslav a Cyril KLIMEŠ. Provozujte IT jinak: agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb. Praha: Grada, 2011. Průvodce (Grada). ISBN 978-80-247-4137-6.

Seznam příloh

Příloha A – 12 principů agilního vývoje softwaru

- *„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 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.“*

Zdroj: [21]