Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Podniková informatika

Přístup DevOps v podmínkách outsourcingu vývoje softwaru

DIPLOMOVÁ PRÁCE

Student : Bc. Martin Dvořák

Vedoucí : doc. Ing. Alena Buchalcevová, Ph.D.

2019

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 20. června 2019 . . . . . .

Bc. Martin Dvořák

Poděkování

Rád bych poděkoval vedoucí diplomové práce, paní doc. Ing. Aleně Buchalcevové, Ph.D., za odbornou pomoc, cenné rady a podnětné připomínky při zpracování této práce. Poděkování dále patří mému vedoucímu v práci, mé manželce a mým dvěma dcerám za podporu nejen v průběhu tvorby diplomové práce, ale i po celou dobu mého studia.

Abstrakt

Diplomová práce se zaměřuje především na přístup DevOps v podmínkách outsourcingu vývoje softwaru. Práce je rozdělena na tři části. V teoretické části je nejprve vysvětlen koncept DevOps, jenž spojuje agilní způsob vývoje software, přístupy Continuous Integration a Continuous Delivery, automatizaci a mnoho dalších osvědčených postupů. Popsány jsou důvody pro jeho zavedení, principy a kultura. Agilní způsob vývoje software je vysvětlen na nejpoužívanějších agilních metodikách Scrum a Kanban. Poté je charakterizován outsourcing IS/ICT a jeho formy, z nichž je dále analyzován outsourcing vývoje softwaru společně s procesem výběrového řízení a smluvními vztahy. Druhá část práce je věnována sloučení obou konceptů na základě analýzy, syntézy a zkušeností autora z oblasti IT provozu. Za prvé byly navrženy zásady, které mají přispět k efektivnímu spolupráci obou partnerů a k neustálému zlepšování v oblastech: spolupráce, kvality produktů (služeb), procesů a organizace, a za druhé bylo navrženo několik vhodných modelů týmové organizační struktury pro sloučení DevOps a outsourcingu vývoje softwaru. Třetí část práce se věnuje ověření zásad na konkrétním softwarovém projektu, v rámci kterého je vývoj aplikace realizován externím dodavatelem. Při zpracování diplomové práce byly využity metody: vědecký popis, explanace, analýza, syntéza, pozorování a dotazování. Hlavními přínosy práce je návrh souboru zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru, jejich charakteristiky, hodnocení, měření a návrh nejvhodnějších modelů týmových organizačních struktur, které jsou využitelné v praxi.

Klíčová slova

DevOps, outsourcing, vývoj softwaru, provoz, agilní

Abstract

The thesis focuses mainly on the DevOps approach under the conditions of software development outsourcing. The thesis is divided into three parts. The theoretical part first explains the concept of DevOps, which combines agile software development, Continuous Integration and Continuous Delivery, automation and many other best practices. The reasons for its implementation, principles and culture have all been described. Agile software development is explained on the most widely used agile methodologies, Scrum and Kanban. Then IS / ICT outsourcing and its forms are characterized, from which the software development outsourcing is analyzed together with the selection process and contractual relations. The second part is devoted to the merging of both concepts based on the analysis, synthesis and experience of the author in the field of IT operations. Firstly, policies have been devised to contribute to the effective cooperation of both partners and to continually improve in the areas of: collaboration, product quality (services), processes and organization; and secondly, several suitable team organizational structure models have been proposed to consolidate DevOps and software development outsourcing. The third part of the thesis deals with the verification of principles on a particular software project, in which the application development is realized by an external supplier. The thesis has been completed using the following methods: scientific description, explanation, analysis, synthesis, observation and questioning. The main benefits of the thesis are the design of a set of principles for DevOps access in terms of software development outsourcing, their characteristics, evaluation, measurement and design of the most suitable models of team organizational structures that can be used in practice.

Keywords

DevOps, outsourcing, software development, operations, agile

Obsah

Úvod

Žijeme v informační společnosti, která je velmi závislá na různých technologických inovacích. Odehrávají se všude ve světě a neustále ovlivňují způsob práce, života i komunikaci lidí. Za těchto podmínek, kdy lidé jsou více propojeni, je jim poskytována široká škála informací a možností v celé řadě oblastí. Na druhou stranu očekávají rychlé a konzistentní informace, jež budou neustále dostupné. Podobně očekávají vedoucí pracovníci obchodních útvarů v podnikatelské sféře od svých IT týmů, že budou schopni stejným způsobem vytvářet příslušné aplikace a programy, které umožní udržovat náskok v konkurenčním prostředí a poskytovat takové produkty a služby, se kterými budou mít zákazníci vynikající zkušenosti a které budou plnit nejen jejich současné, ale i budoucí požadavky.

Informační společnost se postupně vyvíjí ve znalostní společnost. Každá firma již však nemusí obsáhnout všechny vědomosti nutné ke splnění cílů vlastního rozvoje a dosažení skvělých výsledků. Primárně se musí soustředit na své klíčové činnosti, které jí generují zisk, a orientovat své nejlepší lidi na tyto činnosti. Pro ostatní činnosti nesouvisející s vlastním oborem firmy je jednou z možností, jak získat potřebné znalosti a zdroje, nákup služeb u externích poskytovatelů tzv. outsourcing.

Tento vývoj ve znalostní společnost je neoddělitelně spojen s rozmachem informačních a komunikačních technologií, který nutí mnoho společností vyvíjet a provozovat své vlastní softwarové produkty. Ať už pro podporu svých funkcí či jako podporu prodeje nebo pro vytvoření konkurenční výhody a otevření nových tržních možností. Jak tedy skloubit vývoj a provoz softwarových produktů, které nepatří do hlavní činnosti podniku? Odpovědí na tuto otázku může být osvědčený přístup DevOps[1] ve spojení s využitím outsourcingu vývoje a provozu softwaru. Namísto toho, aby podnik softwarové produkty sám vyvíjel a provozoval, najme si na to buď expertní firmu poskytující outsourcing v oblasti IS a technologií, která bude disponovat dostatečnými kapacitami jak v podobě odborníků, tak i v podobě technických či softwarových prostředků a infrastruktury, nebo i externí dodavatele jako jsou programátoři pracující jako živnostníci.

V dynamickém prostředí technologických společností v mezinárodním kontextu se ustálil a dále se prosazuje přístup DevOps jako jedna z nejlepších praxí pro správu IT. Principy přístupu DevOps používají vysoce výkonné technologické společnosti jako Google, Amazon, Facebook, Etsy a Netflix.

DevOps je akronym anglického slovního spojení Development a Operations. Označuje se jím koncept, který usiluje o spojení či určitý stupeň integrace dvou dříve zcela oddělených IT oblastí: vývoje (Dev) a provozu (Ops). Přístup DevOps charakterizuje, že se stále ještě formuje a je otevřený novým myšlenkám, které ho posouvají dále. Jeho principem je dodávka levnějších softwarových produktů ve vyšší kvalitě a rychlosti za pomoci agilních přístupů, které jsou vhodnější pro vývoj softwaru a které umožňují hbité reakce nejen na požadavky ze strany uživatelů a klientů,

[1] DevOps je akronym anglického slovního spojení Development a Operations. Označuje se jím koncept, který usiluje o spojení či určitý stupeň integrace dvou dříve zcela oddělených IT oblastí – vývoje (Dev) a provozu (Ops). (autor)

ale i na hlášené defekty, incidenty a problémy. Dobře zvládnutý DevOps umožňuje nasazovat nové verze softwaru automatizovaně, rychle a spolehlivě.

O obě témata se zajímám, protože pracuji jako aplikační inženýr v IT oddělení v bance, kde se ve velké míře využívá externího vývoje softwaru a kde jsou zároveň uplatňovány některé principy DevOps. V tomto směru se chci podívat na problémy a příležitosti, s kterými je to spojené. Firmy si musí v tomto směru položit řadu otázek, neboť dle Gartner (2017) bude do roku 2020 pocházet 50% až 80% kódu v nových aplikacích od externích třetích stran. Na tuto skutečnost je třeba se připravit a zahrnout ji do strategického plánování. Je však možné zahrnout dodavatele z třetích stran do přístupu DevOps?

Realizace outsourcingu vývoje a implementace softwaru nebo jeho provozu je stále více přijímáno a stává se běžnou věcí. Jeho podíl konstantně narůstá. Zvyšují se jeho možnosti například najmutím vývojového týmu z druhého konce světa. Outsourcing sebou ale přináší i rizika a provozní problémy. Práce by měla odpovědět na to, jak je řešit v kontextu přístupu DevOps.

V oblasti vývoje softwaru se stále více celosvětově prosazují agilní metody a přístupy ve velkém množství softwarových projektů. Jsou více úspěšné a preferují je i vývojové týmy. Čas je cennou komoditou a agilní metodiky urychlují dodávku a tím i nasazení softwaru do produkce. Na druhou stranu je třeba k tomu nastavit podmínky, prostředí, kulturu a organizaci – nejlépe vše založené na principech DevOps.

Zájem o přijetí DevOps proto stoupá mezi firmami, neboť umožňuje účelně a efektivně vynakládat peníze na kvalitní softwarové produkty či služby po celou dobu jejich životního cyklu. Pokud firma přijme DevOps a zároveň si vývoj a implementaci softwaru zajišťuje pomocí externích dodavatelů, je třeba se zabývat následujícími tématy: jak to bude fungovat; co konkrétně je nutné zajistit pro spolupráci, která je klíčová, a pro rychlé nasazování softwaru; jak rozdělit zodpovědnosti a role; jaké problémy lze očekávat a jak měřit úspěšnost spolupráce a výkonnost IT.

Způsob, jakým se přistupuje k řízení externího dodavatele, se velmi liší od tradičních metod. Jednou z otázek, které je třeba si klást, je, jak zajistit, aby měl externí dodavatel společný cíl s objednavatelem. Další z důležitých otázek jsou, jaké jsou možnosti vztahů s externím partnerem a jak bude vypadat týmová organizace.

Smluvní zajištění spolupráce a dodávek softwaru je nezbytné. V této práci se chci zabývat také tím, jaké je nutné formulovat smlouvy pro vývoj a implementaci softwaru, jak je to s autorskými a vlastnickými právy k softwaru, pokud spolupracuje organizace s externími programátory a dodavateli a jaké smluvní záruky je vhodné sjednat pro provoz.

Ve vztahu ke třetím stranám a přístupům do interních systémů, především pak přes externí rozhraní API, je nevyhnutelné se zabývat důvěrou. Dle Gartner (2017) utrpí do roku 2020 50% organizací škody způsobené tím, že se jim nepodaří spravovat důvěru v životní cyklus jejich softwaru.

Z průzkumů, které jsem si provedl při výběru tématu práce, může se potýkat s touto problematikou více firem. Záměrem této práce je prozkoumat DevOps a navrhnout soubor zásad pro to, jak by mělo být přistupováno k implementaci DevOps v podmínkách outsourcingu vývoje softwaru, aby byla zajištěna kvalitní a bezproblémová spolupráce a partnerství v oblasti vývoje a provozu softwaru.

Společnost, která využije outsourcingu vývoje softwarových produktů, se může zcela koncentrovat na své klíčové činnosti, které přinášejí přidanou hodnotu zákazníkovi.

Vymezení tématu práce a důvod výběru tématu

Téma práce jsem si sám zvolil. Inspiraci pro sepsání této práce jsem našel v zaměstnání v bankovním IT prostředí, kde vývoj a provoz pracuje podle některých principů DevOps. Z důvodu toho, že si banka nemůže zajistit vše sama, musí si najímat externí odborníky, outsourcovat nejen dodávku softwarových produktů, ale i využívat aplikačních služeb celé řady IT firem. Outsourcing v oblasti IS/ICT je narůstajícím trendem, společnosti tak řeší své potřeby po informačních technologiích, aby mohli rychle a flexibilně reagovat na požadavky zákazníků. Formou outsourcingu mohou zároveň řešit správu, provoz a údržbu IS a aplikací.

O konceptu DevOps, který mě zaujal, se píše v České republice málo. V mezinárodních IT společnostech ale tento trend znají, DevOps postupně zavádějí a dále ho rozvíjejí. Změna myšlení a firemní kultury, kterou přináší DevOps, umožňuje vysoce výkonné IT a celou řadu inovací a inovace jsou základním předpokladem pro budoucí úspěch v podnikání. V této souvislosti jsem si položil otázku, jak uspořádat vývoj a provoz softwaru na principech DevOps v podmínkách, kdy je vývoj softwaru zajišťován externím dodavatelem formou outsourcingu, aby to vše hladce fungovalo a přinášelo skvělé výsledky. Jak jsem zjistil, třebaže s tím firmy mají obtíže, nikde není popsáno, jak implementovat přístup DevOps v podmínkách outsourcingu a podle čeho se řídit. Tento trend je ale nutné reflektovat a zpracovat. Z tohoto důvodu jsem se v této práci zaměřil na tuto problematiku a rozhodl se formulovat soubor zásad, kterými je třeba se řídit.

Charakteristika současného stavu problémové oblasti

Klíčovou součástí této práce je porozumění oběma doménám – přístupu DevOps a outsourcingu vývoje softwaru.

Mínění IT veřejnosti se posouvá ve prospěch DevOps, technologie cloudů a dalšímu velkému trendu, jímž je outsourcing různých funkčních oblastí včetně analýzy, vývoje, implementace a provozu softwaru. Pro outsourcing IS/IT hovoří jednak to, že poskytování služeb externími dodavateli je zavedený a ve stále větší míře využívaný koncept, pro který existuje rozvinutý trh. Hlavním důvodem je skutečnost, že poskytovatelé outsourcingu IS/IT poskytují širokou škálu nových technologií, kvalitní metodiky a zkušené kvalifikované pracovníky, do kterých dlouhodobě investují. Ve svých službách dosahují vysoké úrovně, neboť se na ně specializovali a zpravidla se na ně zaměřili jako na svou hlavní činnost.

DevOps je relativně nový koncept, pro mnohé stále filozofie, který pomáhá řešit tradiční problémy mezi vývojem a provozem softwaru. Je podporován mezinárodním hnutím IT odborníků, používají ho vysoce výkonné zahraniční společnosti a získává si stále více příznivců. To ho předurčuje stát se paradigmatem pro změny v myšlení, jak má fungovat inovované IT, a s tím související změny firemní kultury a organizace. Jedním z cílů této práce je objasnit, co znamená přístup DevOps.

Silná orientace na služby poskytované zákazníkům nutí podniky soustředit se na technologie virtualizace a Cloud Computingu, který řešily buď prostřednictvím veřejného cloudu od externího poskytovatele, privátním cloudem ve vlastním datacentru nebo jejich hybridními podobami. Po zavedení cloudů je však nezbytné dále uvažovat, jak zefektivnit a zeštíhlit provoz a správu ICT z důvodu tlaku na větší kvalitu, rychlost změn a úspory nákladů.

Cíle práce

Hlavním cílem diplomové práce je formulovat soubor zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru, navrhnout vhodný model týmové organizační struktury a ověřit je v praxi.

Na základě hlavního cíle jsou stanoveny následující dílčí cíle práce:

- Dílčím cílem DC1 je charakterizovat přístup DevOps a s ním spojený agilní vývoj softwaru.
- Dílčím cílem DC2 je analyzovat kulturu, důvody, klíčové principy, procesy od vývoje po provoz softwaru (Průběžná integrace, Plynulé nasazování atd.) a organizační struktury, na nichž je založen holistický přístup DevOps.
- Dílčím cílem DC3 je charakterizovat outsourcing vývoje softwaru jako služby poskytované třetí stranou, jeho přínosy, určení dodavatele, smluvní vztahy a identifikovat rizika spojená s tímto typem outsourcingu.
- Dílčím cílem DC4 je formulovat zásady pro fungování DevOps a outsourcingu vývoje softwaru a identifikovat případná úskalí tohoto spojení.
- Dílčím cílem DC5 je navrhnout pro spojení DevOps a outsourcingu vývoje softwaru jeden nebo více vhodných modelů týmové organizační struktury, stanovit metriky a definovat přístup k důvěře a bezpečnosti mezi organizací a třetí stranou.
- Dílčím cílem DC6 je ověření navržených zásad a postupů DevOps v podmínkách outsourcingu vývoje softwaru v praxi.

Použité metody

Charakter této práce naznačuje, že v prvé řadě bylo nutné provést rešerše informačních zdrojů za účelem získání přehledu o zkoumaných oblastech DevOps a outsourcingu vývoje softwaru. Především o DevOps pocházejí informační zdroje téměř výhradně z odborné zahraniční literatury nebo z odborných článků na zahraničních webech. Pro osvojení si znalostí o DevOps byly vyhledány a vybrány významné publikace z poslední doby, neboť se tato oblast rychle vyvíjí. Pojem outsourcing je veskrze známý, ale k prozkoumání outsourcingu vývoje softwaru bylo nutné vyhledat publikace, které se věnují specifikům tohoto typu outsourcingu.

Za účelem dosažení stanoveného hlavního cíle bylo pro definování a vysvětlení podstaty DevOps a jeho základních částí použito metod vědeckého popisu, explanace a analýzy. Stejným způsobem byly vysvětleny principy agilního vývoje softwaru, na kterých je postaven DevOps, a dvě nejpoužívanější agilní metodiky Scrum a Kanban. Dalším důležitým tématem práce byl popis

a analýza outsourcingu vývoje softwaru pro objasnění jeho specifik a rizik. Kromě toho byl stejnému zkoumání podroben i proces určení dodavatele a související smluvní vztahy.

Na základě analyzovaných oblastí a souvislostí byl metodou syntézy navržen soubor zásad pro úspěšné zavedení přístupu DevOps v podmínkách outsourcingu vývoje softwaru a nejvhodnější modely týmové organizační struktury.

Ověření navržených zásad v kontextu DevOps bylo provedeno na základě jejich uplatnění v softwarovém projektu vývoje aplikace pro generování dokumentů, který je u zákazníka zajišťován externím dodavatelem. Popsané skutečnosti byly získány metodami pozorováním a dotazováním se členů vývojového a testovacího týmu, zástupců byznysu a projektového manažera. Při zpracování práce a plnění cílů byly oporou znalosti a zkušenosti získané dlouholetou prací v oblasti provozu IS/ICT.

Přínosy práce

Mezi očekávané přínosy této práce patří:

- Charakteristika přístupu DevOps a s ním spojených, nejpoužívanějších agilních metodik vývoje softwaru.
- Shrnutí klíčových principů, kultury a špičkových prediktorů výkonnosti IT [2]v DevOps.
- Shrnutí holistického přístupu DevOps, který je postaven i na procesech, které byly do něj integrovány a staly se jeho součástí, i když původně vznikly jako nezávislé přístupy (Průběžná integrace, Plynulé nasazování a tak dále).
- Zhodnocení přínosů, rizik a další důležitých aspektů outsourcingu vývoje softwaru jako služby poskytované třetí stranou.
- Návrh souboru zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru včetně jejich charakteristiky, hodnocení a měření.
- Stanovení metrik pro měření výkonnosti IT a definice přístupu, jak zajistit důvěryhodnost mezi organizací a třetí stranou poskytující outsourcing vývoje softwaru
- Návrh jednoho a více nejvhodnějších modelů týmových organizačních struktur pro přístup DevOps v propojení s outsourcingem vývoje softwaru.
- Ověření použitím navržených zásad a postupů DevOps v podmínkách outsourcingu vývoje softwaru na vývoji a provozu aplikace v bankovním prostředí.

Cílová skupina

Tato práce je určena nejen pro rozhodující orgány (tzv. decision makers) ve firmách - výkonné manažery, vedoucí pracovníky IT oddělení, projektové manažery, ale i pro vývojáře, analytiky a testery z IT firem, provozní inženýry a zákazníky. Rovněž může být i zajímavá pro odborníky z dodavatelských firem poskytující outsourcing v IS/IT a pro studenty.

[2] Prediktory výkonnosti IT jsou nástroje, procesy, pravidla nebo kultura, které předpovídají, že se IT v budoucnu stane více výkonným, nebo to bude jejich důsledkem. Viz také kapitola *Terminologický slovník*.

1 Rešerše informačních zdrojů

Tato kapitola se zaměřuje na přehled odborných publikací, akademických prací a odborných článků z oblastí konceptu DevOps a outsourcingu, které byly podstatné pro tuto diplomovou práci.

K DevOps jsou informace decentralizované, chybí zatím centrální zdroj informací. Ohledně praktik DevOps lze rovněž konstatovat, že chybí standardizace. Formální definice DevOps je dostupná jen v zahraniční odborné literatuře.

Jedna z nejzajímavějších publikací o DevOps z poslední doby je publikace od kolektivu autorů kolem Kima Gene *The devops handbook: how to create world-class agility, reliability, & security in technology organizations* (Kim et al. 2016).

Na odborných článcích o DevOps se významnou měrou podílí celá řada organizací a osobností hnutí DevOps. Na straně organizací lze jmenovat asociaci DevOps Agile Skills Association (DASA) a jejich odborné články (tzv. white papers) jako je článek *Embracing Digital Disruption by Adopting DevOps Practices* (Farenhorst et al. 2016), organizaci ISACA a jejich sérii článků o DevOps počínaje shrnutím DevOps v článku *DEVOPS OVERVIEW* (ISACA, 2015) nebo společnost Puppet, která každý rok provádí průzkum stavu DevOps v organizacích a zveřejňuje o něm reporty. Pro rok 2018 to byl report *2018 State of DevOps Report* (Puppet Labs, 2018).

Problematikou důvěryhodnosti ve vztahu k třetím stranám se zabývá odborná studie *Managing Digital Trust in the Software Development Life Cycle* (Gartner, 2017) od společnosti Gartner, která se zabývá výzkumem a poradenstvím v oblasti IS/ICT a technologií.

Odborných knih a akademických prací o outsourcingu je hodně, ale publikací o outsourcingu pro oblast IS/ICT mnoho není. S ohledem na cíle práce a zkoumanou problematiku jsem pročetl publikace od autorských týmů z kolektivů VŠE. Jedná se zejména o publikace *Tvorba informačních systémů* (Bruckner et al. 2012) a *Aplikační služby IS/ICT formou ASP: proč a jak pronajímat informatické služby* (Voříšek et al. 2004), dále pak o starší, ale v zásadě stále platnou e-knihu *Outsourcing a jeho aplikace při řízení informačního systému podniku*.

Tak jako v jiných oblastech jsou i vztahy mezi nezávislými subjekty v oblasti IS/ICT legislativně regulovány. Oblast IT má však svá specifika, která vyžaduje specializované právníky. Právním specifikům této oblasti se věnuje publikace *Softwarové právo* (Jansa et al. 2018), ve které autoři shrnuli vše podstatné z této problematiky a poskytli řadu doporučení.

1.1 Odborné knihy

Publikace od kolektivu autorů kolem Kima Gene *The devops handbook: how to create world-class agility, reliability, & security in technology organizations* (Kim et al. 2016) představuje, jak název napovídá, příručku pro implementaci prvotřídního DevOps v technologických společnostech.

Další velmi dobře strukturovaný materiál o DevOps a jeho agilních základech je licencovaný eBook *DevOps Fundamentals* organizace DevOps Agile Skills Association, akreditováno ITpreneurs, edice od společnosti GOPAS (DevOps Agile Skills Association, 2017). Jedná se o první z řady

certifikovaných materiálů organizace DevOps Agile Skills Association, který pojednává o základech, na kterých je postaven DevOps.

Tématu právního prostředí pro oblast IT a softwarového práva v České republice se věnuje publikace *Softwarové právo* od zkušených advokátů Lukáše Jansy, Petra Otevřela a Martina Števka (Jansa et al. 2018). V této publikaci se řeší otázky formulace smluv a práva v oblasti IT se zaměřením na tvorbu, dodávku a servis softwaru s cílem se vyhnout právním rizikům. Obsahuje mnoho praktických doporučení a je cenným zdrojem poznatků v oblasti IT práva zaměřujícího se na tvorbu, dodávku a servis softwaru. Zahrnuje mimo jiné právní pojetí outsourcingu a Cloud computingu a další témata.

Aspekty tvorby informačních systémů od analýzy k návrhu až po provoz se zabývá kolektiv uznávaných autorů v publikaci *Tvorba informačních systémů: principy, metodiky, architektury* (Bruckner et al. 2012). Věnují se na jedné straně vztahům mezi byznysem a IS/ICT a na druhé metodickým přístupům. Především pak metodice MMDIS. Autoři dále v publikaci nastiňují možnosti outsourcingu IS/ICT.

Jedné z forem outsourcingu, vzdálenému pronájmu aplikací přes internet (ASP), se věnuje publikace s názvem *Aplikační služby IS/ICT formou ASP: proč a jak pronajímat informatické služby* (Voříšek et al. 2004). Kolektiv autorů kolem Jiřího Voříška v ní uceleně popisuje fungování modelu ASP, vztahy mezi zákazníkem a poskytovatelem včetně dohody SLA. Nicméně neobsahuje nic o Cloud computingu.

1.2 Akademické práce

Diplomová práce od Karla Slatinského *Integrace agilních a DevOps přístupů do ITIL* (Slatinský, 2016) se zabývá možnou integrací agilních přístupů metodiky SCRUM a zastřešující metodiky DevOps do životního cyklu služby dle knihovny ITIL a případnými přínosy tohoto propojení.

1.3 Odborné články

Odborná studie od společnosti Gartner *Managing Digital Trust in the Software Development Life Cycle* (Gartner, 2017) se zabývá posuzováním a klasifikací důvěryhodnosti z různých aspektů při tvorbě aplikací třetími stranami v nastupujícím digitálním světě.

Odborný článek (tzv. white paper) *DEVOPS OVERVIEW* ze série „An ISACA DevOps Series White Paper“ od sdružení ISACA poskytuje ucelený přehled o DevOps (ISACA, 2015). Srozumitelně shrnuje vznik hnutí DevOps, agilní vlivy, důvody, faktory úspěchu pro úspěšnou implementaci, dopady, přínosy pro byznys, výzvy a v neposlední řadě předkládá vizi DevOps s ohledem na potřebné dovednosti, kulturu a procesy.

Odborný článek od autorů Rika Farenhorsta a Nielse Loadera, který se řadí mezi odborné články (tzv. white papers) asociace DevOps Agile Skills Association (DASA), s názvem *Embracing Digital Disruption by Adopting DevOps Practices* (Farenhorst et al. 2016) pojednává o potenciálu DevOps, principech, odborné způsobilosti pro DevOps a rámci kompetencí pro DevOps, který identifikuje osm oblastí znalostí a čtyři dovednosti relevantní v DevOps.

Směřování a stav DevOps na mezinárodní úrovni zkoumá společnost Puppet Labs, která každý rok vydává zprávu o stavu DevOps. V této práci byly použity celkem reporty z let 2014, 2016, 2017 a 2018: *2014 State of DevOps Report* (Puppet Labs, 2014), *2016 State of DevOps Report* (Puppet Labs, 2016), *2017 State of DevOps Report* (Puppet Labs, 2017) a *2018 State of DevOps Report* (Puppet Labs, 2018). Obsahem těchto reportů plných zajímavých informací jsou pravidelně nejen klíčové poznatky z proběhlých výzkumů, výsledky výkonnosti IT a výkonnosti organizací či spokojenosti zaměstnanců, ale i různá aktuální témata pro daný rok jako například Lean management produktů, změna organizační kultury a identity, transformační vedení lidí a evoluce DevOps na základě měření CAMS (kultury, automatizace, měření a sdílení).

2 DevOps

Tato kapitola má za cíl poskytnout čtenáři základní přehled o fenoménu DevOps, který v současné době rezonuje v odvětví IS/ICT a který přináší příležitost ke změně oproti tradičnímu IT. V běžném pojetí IT vede často spolupráce byznysu, vývojářů a odborníků z IT provozu k nedorozuměním a konfliktům. Z tohoto důvodu v mnoha lidech vyvolává pocity beznaděje a zoufalství.

Popisem přístupu DevOps a agilního způsobu vývoje softwaru naplňuje tato kapitola dílčí cíl práce **DC1.** Zároveň naplňuje dílčí cíl práce **DC2** prostřednictvím analýzy a popisu kultury, důvodů, klíčových principů, integrovaných přístupů a organizační struktury, na nichž je založen přístup DevOp,**.** Dále slouží tato kapitola jako podklad pro kapitolu 4, ve které jsou blíže popsány navržené zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru.

DevOps je možné charakterizovat jako filozofii, hnutí či změnu kultury, která zdůrazňuje spolupráci a komunikaci jak vývojářů softwaru, tak i dalších odborníků v oblasti informačních technologií (IT). V současné době přistupují lidé k DevOps rozdílnými způsoby, podle toho, jak mu rozumí a vykládají si jeho různé aspekty a rozsah. Z tohoto důvodu je nutné pro účely této práce smysluplně definovat a popsat, co znamená přístup DevOps a jaký má vliv na vývoj a provoz v oblasti IS/ICT.

Stejným způsobem je nutné vymezit agilní způsob vývoje softwaru, který je nedílnou součástí DevOps a dalších integrovaných přístupů a který je nezbytný také pro zásady pro úspěšnou spolupráci v rámci outsourcingu vývoje softwaru.

Následující podkapitola charakterizuje modely životního cyklu vývoje softwaru a slouží jako podklad pro následný popis agilního způsobu vývoje softwaru v kapitole 2.2.

2.1 Životní cyklus vývoje softwaru

Životní cyklus vývoje softwaru vychází ze softwarového inženýrství a definuje opakovaný proces vývoje softwaru, který se skládá ze stanovených etap. Z hlediska postupu se rozdělují životní cykly vývoje na tři základní typy:

- Vodopádový model
- Iterativní vývoj
    - Inkrementální model
    - Evoluční model

Tradiční přístup k vývoji softwaru zaujímá vodopádový model, ve kterém postupuje po fázích, od jedné fáze do další fáze. Mezi fázemi se nelze vracet. Skládá se z fází: specifikace požadavků, analýza, návrh, implementace, testování a zavedení. Tento model se používá do dnes z důvodu své jednoduchosti.

Iterativní model řeší slabiny tradičního přístupu. Iterativní model je především spojen s inkrementálním přístupem. Tento model charakterizuje to, že před zahájením vývoje je možné definovat požadavky jen v hrubé formě. V průběhu iterací se pak řeší části požadovaného produktu

(přírůstky) ve spolupráci se zákazníkem. Výsledkem každé iterace je spustitelná verze, na které si může zákazník ověřit, že se řešení vyvíjí požadovaným směrem a poskytnout zpětnou vazbu řešiteli. Tento přístup umožňuje snazší zapracování změn požadavků. Produkt je kompletován postupně po přírůstcích, které jsou výstupem jednotlivých iterací. Další výhodou je rovnoměrnější pracovní vytížení vývojářského týmu.

V evolučním (adaptačním) modelu naproti tomu nejsou na začátku známy všechny požadavky. Zákazník je definuje postupně a to na začátku každé iterace.

V současné době nelze opomenout skutečnost, že životní cyklus vývoje softwaru prochází výraznou proměnou a stále více se prosazuje iterativní a přírůstkový proces vývoje softwaru s využitím Lean praktik. Pomocí tohoto přístupu dokáží týmy dodávat průběžně části produktu zákazníkovi k upřesnění jeho požadavků, jakož i pružně a efektivně reagovat na změny požadavků. Z tohoto důvodu se v čím dál větší míře uplatňují různé varianty agilních rámců. V souladu s cíli této práce se tato kapitola zaměří na způsob agilního vývoje softwaru a agilní rámce Scrum a Kanban, jejichž hlavním motivem je dodání softwaru s důrazem na předání ve stanoveném čase. Základním předpokladem pro tuto práci je to, že organizace zná své procesy životní cyklu vývoje softwaru a má je popsány.

2.2 Agilní vývoj softwaru

Pro vývoj softwaru se celosvětově osvědčily agilní přístupy. Mají větší úspěšnost v dodávce softwaru než tradiční modely vývoje softwaru jako je vodopádový model.

Ve vztahu k cílům práce musím konstatovat, že DevOps je postaven právě na agilních přístupech, a proto se budu v této práci zabývat principy a základními metodikami agilního vývoje softwaru. K základním rysům těchto metodik patří to, že lépe reagují na počáteční neúplné zadání požadavků na softwarové řešení a na často se měnící požadavky zákazníků, kteří v mnoha případech podnikají v prostředí vyžadující provádění neustálých změn.

Stěžejním principem agilního pojetí je to, že je uspokojován zákazník průběžným dodáváním softwarového produktu nebo služby, přičemž se sám pravidelně podílí na vývoji produktu v průběhu a na závěr sprintu při demo produktu. Tato zásada zviditelňuje to, co má být dodáno. Díky využití zpětné vazby od zákazníka se vývojový tým začne včas pohybovat správným směrem a tvořit správný produkt. Na konci každého sprintu je pravidelně dodáván potenciálně způsobilý produkt, který lze využít k vytváření hodnoty pro byznys po celý cyklus vývoje softwaru. Nejdůležitější funkce jsou dodány okamžitě a neváznou v cyklu vývoje. Na druhou stranu vyhovují agilní přístupy více i vývojářům ve srovnání s přístupy založenými na tradičních obsáhlých (rigorózních) metodikách. Dávají jim větší prostor na práci, smršťují byrokracii na minimum, zároveň ale vynucují alespoň určitou míru komunikace mezi lidmi. Za tímto účelem zavádějí důležitou průběžnou sebereflexi, a to jak produktu, tak i způsobu, jakým je produkt vytvářen. Tato praktika umožňuje učit se z chyb a neustále dělat věci lépe. Pokud jsou agilní metodiky soustavně správně praktikovány, má to za následek u lidí vyšší angažovanost a spokojenost s prací.

Agilních metodik je hodně. Nejde ale o to, jakou přesně používáme agilní metodiku. Jde o to, jestli se tým drží zásad, které tyto metodiky mají v sobě. Pokud chce tým agilně vyvíjet software, musí

ctít bez výjimky všechny principy agilního vývoje softwaru deklarované v tzv. Agilním Manifestu (anglicky Agile Manifesto) a řídit se jimi. Pokud se řídí jen některými principy a ostatní ani nezná, nemůže být tým považován za agilní.

V Agilním Manifestu (Agile Alliance, 2001) jsou zdůrazněny čtyři body, které jsou považovány za ty, které pomáhají k objevování lepších způsobů vývoje softwarových produktů a na které je potřeba se zaměřit. Jedná se o jednotlivce a interakce mezi lidmi, fungující software, spolupráci se zákazníky a reagování na změny.

Pro úplnost uvádím celé znění Agilního Manifestu:

> ***„Manifest Agilního vývoj softwaru***
>
> *Objevujeme lepší způsoby vývoje software 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.*“ (Agile Alliance, 2001)

Autoři Manifestu Agilního vývoje dále vyhlásili, že se řídí následujícími dvanácti základními principy:

- 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 často, v intervalech týdnů až měsíců, s preferencí kratšího období.
- Lidé z byznysu a vývojáři 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 postupu vpředu je fungující software.
- Agilní procesy podporují udržitelný vývoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet trvale neměnné tempo.
- Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
- Jednoduchost, tzn. umění maximalizovat množství nevykonané práce, je klíčová.
- Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.

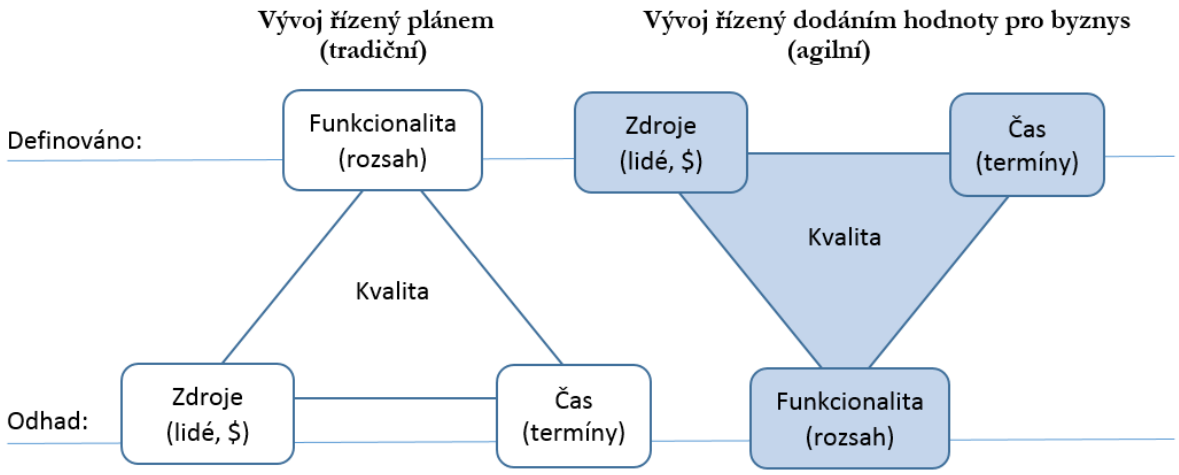
- Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

Výše uvedené principy jsou jádrem všech agilních metodik.

V kontextu projektového řízení se agilní vývoj liší oproti tradičním přístupům, při kterých je vývoj softwaru řízen na základě plánu. Pro tradiční projekty vývoje softwaru je nutné na počátku znát funkční požadavky a mít k nim specifikaci. Při plánování se pak odhaduje, kolik je potřeba lidí, tedy zdrojů, a času pro realizaci požadované funkcionality. Při vývoji se tým řídí plánem.

Naproti tomu v rámci agilního vývoje je to jinak, jak je znázorněno na obrázku *2.1 - Porovnání parametrů vývoje řízeného plánem (tradičního) a vývoje řízeného dodáním hodnoty pro byznys (agilního)*. Při něm se vývojový tým soustředí na dodání produktu či služby (hodnoty pro byznys). Pro agilně řízené projekty vývoje softwaru je typické, že na začátku mám lidi a definovaný čas. Znám kapacitu alokovaných vývojářů a v kolika sprintech má být produkt dodán. Tým dodá produkt, ale ještě neví přesně co. Funkcionalita se určuje ve spolupráci se zákazníkem v průběhu vývoje produktu. Při vývoji se tým řídí dodáním hodnoty pro byznys.

Obrázek 2.1 – Porovnání parametrů vývoje řízeného plánem (tradičního) a vývoje řízeného dodáním hodnoty pro byznys (agilního)



Zdroj: vlastní zpracování (Leffingwell, 2007)

2.2.1 Scrum

V této kapitole je vysvětlen jeden z nejznámějších a nejspolehlivějších agilních rámců v oblasti řízení vývoje softwaru, který vychází ze základních principů Agilního Manifestu. Lze předpokládat, že právě tento agilní rámec budu použit díky své známosti pro vývoj softwarových produktů v podmínkách zapojení externích dodavatelů.

Scrum je agilní metodika řízení vývoje software, která podporuje iterativní a přírůstkový vývoj softwarových produktů a která poskytuje soubor nástrojů, které pomohou být agilními při dodávce softwarových řešení. Scrum je velice dobře popsán a existují k němu praktické návody i oficiální Scrum Guide, který spravují spoluautoři Scrumu Jeff Sutherland a Ken Schwaber. Vývoj softwarových produktů je zpravidla řízen jako projekt. Proto lze říci, že se Scrum používá k agilnímu řízení projektů.

Vývojový proces ve Scrumu je především empirický a odlehčený za účelem dosažení maximální efektivity. V rámci toku Scrum zavádí tři týmové role, šest událostí, tři artefakty, dva stavy (pravidla), tři společenské objekty a v neposlední řadě cykly zlepšování (DevOps Agile Skills Association, 2017).

2.2.1.1 Role ve Scrum týmu

Scrum tým se skládá z Vlastníka produktu, Vývojového týmu a Scrum Mastera. Scrum týmy jsou samo-organizující se a multidisciplinární, tj. s pokrytím více různými funkcemi (Schwaber a Sutherland, 201, s. 67). Mohou mít pět až deset lidí. Každá role ve složení týmu má své opodstatnění a je nedílnou součástí celého Scrum týmu.

Scrum týmy dodávají produkty iteračně a přírůstkově, čímž se maximalizuje počet možností na zpětnou vazbu. Přírůstkové dodávaní "Hotového" produktu zaručuje, že potenciálně použitelná verze funkčního produktu je vždy dostupná (Schwaber a Sutherland, 2017, s. 6).

- Vlastník produktu (anglicky Product Owner) je držitel vize a mise produktu a je odpovědný za maximalizaci hodnoty produktu, který vytváří Vývojový tým. Tuto roli zastává zpravidla osoba, která je kompetentní a zná velmi dobře byznys a zákazníka organizace. Od organizace má zplnomocnění zastupovat v týmu konečné uživatele a zákazníka.
  Vlastník produktu je jedinou osobou zodpovědnou za prioritizaci a seřazení Produktového backlogu a definici uživatelských příběhů, které jsou pro organizaci a zákazníka důležité, a které mají být přednostně vytvořeny. Odkládá další méně důležité požadavky. Zajišťuje, aby požadavky dodržovaly definici připravenosti (DoR) a byly srozumitelné všem v potřebné hloubce. Stará se také o to, aby Produktový backlog byl transparentní a zobrazoval to, na čem bude Scrum tým dále pracovat (Schwaber a Sutherland, 2017, s. 6). V některých případech může být zastoupen i zplnomocněnými Business analytiky.
- Vývojový tým: Multidisciplinární tým složený z odborníků, kteří pracují na úkolech dohodnutých na začátku sprintu s cílem dodat přírůstek produktu dle definice „Hotového“ na jeho konci. Vývojový tým je strukturovaný a zplnomocněný organizací, aby se sám mohl organizovat a řídit svojí práci, jakož i průběžně zlepšovat svůj vlastní proces. Tým je multidisciplinární a to znamená, že má všechny schopnosti potřebné na vytvoření Přírůstku produktu. Typický vývojový tým má velikost čtyři až devět členů (Schwaber a Sutherland, 2017, s. 7).
- Scrum Master: je zodpovědný za propagaci a podporu Scrum procesu. Dělá to tak, že pomáhá všem pochopit teorii, postupy, pravidla a hodnoty Scrumu. Soustřeďuje se na proces a cíle a zajišťuje, že všichni hrají podle pravidel. Scrum Master je na jedné straně služebník a na druhé vůdce týmu Scrum. Osobám mimo tým Scrumu pomáhá pochopit, které z jejich interakcí s týmem Scrum jsou užitečné a které nejsou. Zároveň pomáhá všem změnit tyto interakce, aby maximalizoval hodnotu vytvářenou týmem Scrum (Schwaber a Sutherland, 2017, s. 7).

2.2.1.2 Opakující se události

Scrum je třeba pojímat jako proces, který se dle DevOps Agile Skills Association (2017, s. 122-123) skládá z následujících šesti opakujících se událostí (Schwaber a Sutherland, 2017, s. 9) :

- Sprint: Představuje základní iteraci ve Scrumu, což je krátký časový úsek, který má sloužit k rozdělení práce na menší, dokončitelné a dodatelné celky. Zpravidla je délka sprintu definována jako týden nebo dva týdny, ale může být i delší, ve kterém je třeba dokončit

určitou část softwaru. Důležité je, jak to vyhovuje týmu či produktu. Usiluje se o přirozený rytmus práce a přemýšlení o procesu. Na konci musí být hotová část díla, která se dá předvést zákazníkovi.

- Plánování sprintu nebo také sezení plánovacího pokeru (anglicky Planning Poker Session): Probíhá na počátku každého sprintu za účelem určení cíle sprintu a vytvoření plánu práce, která má být udělána ve sprintu. Vstupem je Produktový backlog, ze kterého vybere vývojový tým pro sprint konkrétní uživatelské příběhy, respektive funkčnosti, zanalyzuje je a případně rozdělí na úkoly, které si rozdělí v týmu. Přitom hraje plánovací poker, aby kvantifikoval množství práce, která je nutná ke splnění vybraných uživatelských příběhů a úkolů. Ohledně odhadů pracnosti (nacenění) se shoduje celý tým a na jejich základě potvrzuje realizaci úloh ve sprintu. V průběhu času po provedení více pokerů se odhady práce stávají spolehlivějšími.
- Denní Stand-Up schůzka (anglicky Daily Scrum): Každodenní pevně daná praktika, která je určena pro vývojový tým a která umožňuje monitorovat stav, plán a překážky softwarového projektu. Každý člen vývojového týmu vysvětlí, co udělal od předchozího dne, kde je teď, co se chystá dělat v daný den a jaké vidí překážky pro svou práci. Pokud jsou přítomny i jiné osoby, Scrum Master zajišťuje, že schůzku nenarušují. Stand-Up by nikdy neměl trvat déle než 15 minut.
- Vyhodnocení sprintu (anglicky Sprint Review), rovněž známé jako demo produktu (anglicky Product Demo): Každý sprint se uzavírá aktivitou demo produktu, které se účastní tým, Vlastník produktu a zákazník. Účast na demo produktu je nezbytná pro zlepšení spolupráce. Je to způsob, jak poskytnout a obdržet zpětnou vazbu od všech zainteresovaných stran a nejen ji zapracovat do produktu, ale i ji využít ke zlepšení procesu během příští iterace.
- Týmová retrospektiva (anglicky Sprint Review): Důležitý aspekt Scrumu, kdy tým vyhodnotí po každém sprintu, co se dělo dobře a co se naopak nedařilo, respektive zkušenosti z proběhlého sprintu. Retrospektiva vytváří bezpečnou zpětnou vazbu a musí proto být kvalitní, strukturovaná a pochopená. Je silným nástrojem pro neustále zlepšování způsobu práce a tvorbu týmových procesů, které týmu ulehčují komplexní fungování.
- Sezení ke zpřesnění backlogu (The Backlog Refinement Session): Praktika uskutečňovaná v půlce sprintu, která slouží k předvídání a určení toho, jaké uživatelské příběhy se očekávají v příštím sprintu. V případě nejasných uživatelské příběhů dává čas vlastníkovi produktu, aby do dalšího sprintu vylepšil uživatelské příběhy.

2.2.1.3 Artefakty

Metodika Scrum předepisuje následující tři artefakty (Schwaber a Sutherland, 2017, s. 15-17):

- Produktový backlog (anglicky Product Backlog): Neustále se vyvíjející, uspořádaný a prioritizovaný seznam funkčních požadavků, tedy všeho, co je zákazníkem v produktu požadováno pro zajištění jeho optimální hodnoty. Jen na základě toho seznamu se smí provádět úpravy produktu. Položky v Produktovém backlogu se jmenují uživatelské příběhy (anglicky User Stories) a musejí splňovat definici připravenosti (DoR). Uživatelské příběhy jsou tedy popisem požadované funkčnosti (co, pro koho a jak) a rozdělují se na jednotlivé úkoly (Tasks). Za Produktový backlog včetně obsahu, dostupnosti a uspořádání požadavků je zodpovědný Vlastník produktu.

- Sprint Backlog: Pro sprint vybraná sada dosud nevyřízených položek z Produktového backlogu společně s plánem na dodání přírůstku a tím dosažení cíle sprintu. Sprint backlog je ze strany vývojového týmu kvalifikovaný odhad realizovatelné funkcionality, která bude součástí dodávky přírůstku produktu, a práce potřebné na přeměnu dané funkcionality na funkční přírůstek na konci sprintu dle definice hotového (DoD), například činnosti jako je kódování, sestavení, kontrola a testování.
- Přírůstek (anglicky Increment): DevOps Agile Skills Association (2017, s. 265) definuje tento artefakt jako potenciálně způsobilý produkt k dodání (anglicky Potentially Shippable Product), tj. hodnotu pro byznys. Jedná se o přírůstek produktu, který je dodán na konci každého sprintu a obsahuje hotový vývoj všech požadavků od předchozího releasu. Pokud by to byznys chtěl, mohl by být produkt nasazen do produkčního prostředí.

2.2.1.4 Stavy

Stavy jsou v metodice Scrum pravidla ulehčující a zrychlující práci. Ve Scrum existují následující pravidla (DevOps Agile Skills Association, 2017, s. 121):

- Definice Připravenosti (anglicky Definition of Ready - DoR): Seznam pravidel popisující standardy, které musí dodržovat uživatelský příběh, aby mohl být přijat týmem vývoje. DoR zajišťuje, že požadavky jsou jasné od svého vzniku a to eliminuje potřebu diskusí během sprintu. Vlastník produktu zajišťuje, aby uživatelské příběhy dodržovaly DoR.
- Definice Hotového (anglicky Definition of Done - DoD): Seznam kritérií popisující, která témata je třeba řešit, aby mohl být produkt považován za potenciálně způsobilý k dodání. Jedná se o jednoduchý seznam obsahující omezení, jako je kódování, unit testy, funkční testy, testy výkonu, akceptační testy (UAT), kontrola a dokumentace. Jasně definuje značku splnění. Tým dodává pouze tu část produktu, která splňuje kritéria v seznamu.

2.2.1.5 Nástroje pro týmovou spolupráci

Scrum tým může použít ke svému řízení a k zachování transparentnosti následující nástroje pro týmovou spolupráci (DevOps Agile Skills Association, 2017):

- Scrum tabule: Na tabuli se vizuálně zaznamenává seznam úkolů, které mají být dokončeny v aktuálním sprintu. Tato tabule může mít podobu například Kanban tabule, kde se úkoly posouvají zleva doprava přes sloupce z "To do", "Doing", "Done" nebo "Impeded".
- Burn Down graf: Během sezení plánovacího pokeru jsou funkcím přiřazeny rychlostní body. Celkové body všech funkcí, které jsou součástí dodávky přírůstku ve sprintu, jsou přidány do Burn Down grafu. Kdykoli je funkce (User Story) dokončená, body "spálené" při práci se odečtou od celkového počtu. Cílem každého sprintu je skončit vlevo vždy s počtem bodům rovným nule. Burn Down graf znázorňuje rychlost spalování v čase za běh sprintu. Ve výsledku pomáhá týmu dosáhnout požadovaného pokroku.
- Tabule překážek (anglicky Impediment Board): Obsahuje všechna témata, která týmu zabraňují v práci. Zpravidla Scrum Master určuje, jak si s překážkami poradit.

2.2.1.6 Tok v metodice Scrum

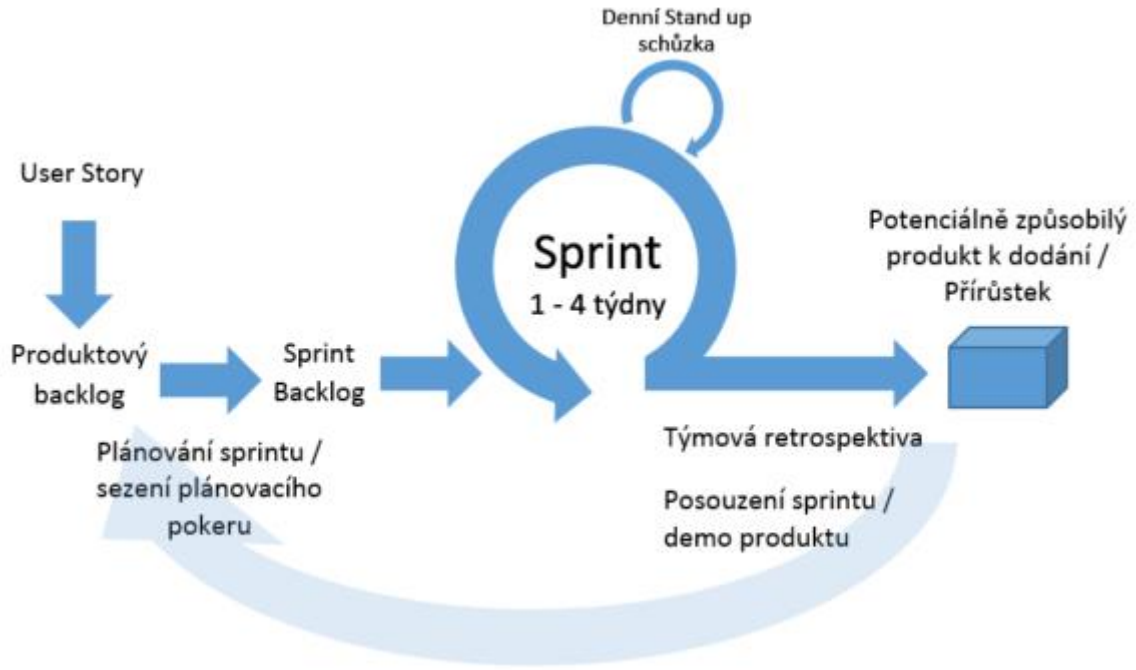
Cílem této části je poskytnout na vysoké úrovni přehled toku ve Scrum procesu, pro něž jsou výchozími předpoklady, že je připraven Produktový backlog, je určen Scrum master a vývojový tým a panuje shoda nad tím, které dokumenty a artefakty se mají dodávat v rámci jednotlivých

sprintů a které až na konci dodávky. Tok ve Scrum procesu začíná nastavením společného cíle pro sprint, je označen zelenými šipkami na následujícím obrázku *2.2 - Tok ve Scrum procesu.*

Kroky ve Scrum procesu jsou (DevOps Agile Skills Association, 2017, s. 123):

1. Společně s vlastníkem produktu definuje tým cíl pro další sprint.
2. Jakmile je cíl stanoven, přistoupí tým k plánování, aby se odhadla pracnost uživatelských příběhů a určil počet příběhů pro sprint.
3. Úlohy, které odpovídají kapacitám vývojového týmu pro sprint, dodržují pravidla DoR a jsou týmu jasné, jsou přidány do sprintu.
4. Sprint je zahájen a vývojáři budou pracovat nepřerušovaně na dohodnutých úkolech. Vlastník produktu nesmí aktualizovat uživatelské příběhy či úkoly v průběhu sprintu.
5. Na konci sprintu při demo produktu je zákazníkům předveden přírůstek produktu.
6. Po předvedení produktu se provádí přezkoumání sprintu a retrospektiva, která umožňuje zlepšení procesu díky zpětné vazbě.
7. Zpřesnění backlogu pomáhá Vlastníkovi produktu, aby uživatelské příběhy byly ve stavu DoR. Tuto činnost lze také provést uprostřed sprintu, pokud je to nutné.
8. Vlastník produktu přidává aktualizované uživatelské příběhy do Produktového backlogu.

Obrázek 2.2 – Tok ve Scrum procesu



Zdroj: vlastní zpracování (DevOps Agile Skills Association, 2017, s. 124).

Pro efektivní využití Scrum je třeba, aby se porozumělo výše uvedeným pojmům a obsahu. Je potřeba si dát pozor, aby se používaly všechny předepsané prvky metodiky Scrum.

2.2.1.7 Cykly zlepšování

Představují způsob, jak zlepšit týmovou práci. Cyklicky během každé z demo relací produktu je možné se zaměřit na zlepšování časového průběhu, produktu a spolupráce, což má pozitivní dopad Product Backlog. Další zlepšovací cyklus umožňuje technika retrospektivy, kterou tým může využít

k postupnému zlepšování procesu tím, že diskutuje o tématech ke zlepšení (DevOps Agile Skills Association, 2017).

2.2.2 Kanban

Kanban je rámec, který byl původně vyvinutý pro zdokonalování systému řízení výrobní logistiky, které říká co, kdy a jak produkovat. Pro své nesporné kvality zeštíhlení (anglicky Lean) řízení a toku materiálu ve výrobě byl převzat do různých oblastí od logistiky přes marketing, lidské zdroje, výzkum až po oblast vývoje softwarových produktů, jehož organizace práce se dá přirovnat k procesu malosériové výroby, která je řízená požadavky zákazníků.

Kanban jako metoda může sloužit jako účinný procesní nástroj zeštíhleného a agilního přístupu, který umožňuje softwarový projekt řídit jednoduše, rychle a transparentně s důrazem na dodání včas. Jak napovídá slovo Kanban, které v japonštině znamená štítek, destička, důležité jsou kartičky, na které se píší úkoly, požadované funkce na základě požadavků zákazníka.

Hlavním znakem Kanban převzatým z výroby je systém tahu (anglicky Pull), v rámci kterého se pracovní zásoba určuje skutečnou potřebou zákazníka.

K řízení práce pomocí kartiček se používá jejich vizuální prezentace na Kanban tabuli, která má v nejjednodušší podobě tři sloupce: "To-Do" (zásobník práce), "Ongoing" (rozpracované úkoly) a "Done" (dokončené úkoly). Možné je ale přidat i další sloupce jako "Integrate", "Test", "Release" či je pojmenovat i jinak Backlog, Ready, Coding, Testing, Approval, and Done (Kniberg a Skarin, 2010, s. 15). Sloupce na tabuli představují etapy celého vývojového procesu. Kartičky na tabuli pak vizuálně ukazují, jak se pohybuje tok práce zleva doprava. Přesun kartiček je závislý na odvedené práci. Kartičky přesouvají členové vývojového týmu podle stavu každé pracovní položky z jednoho sloupce do druhého doprava, aby v reálném čase ukázali pokrok a koordinovali své úsilí s ostatními.

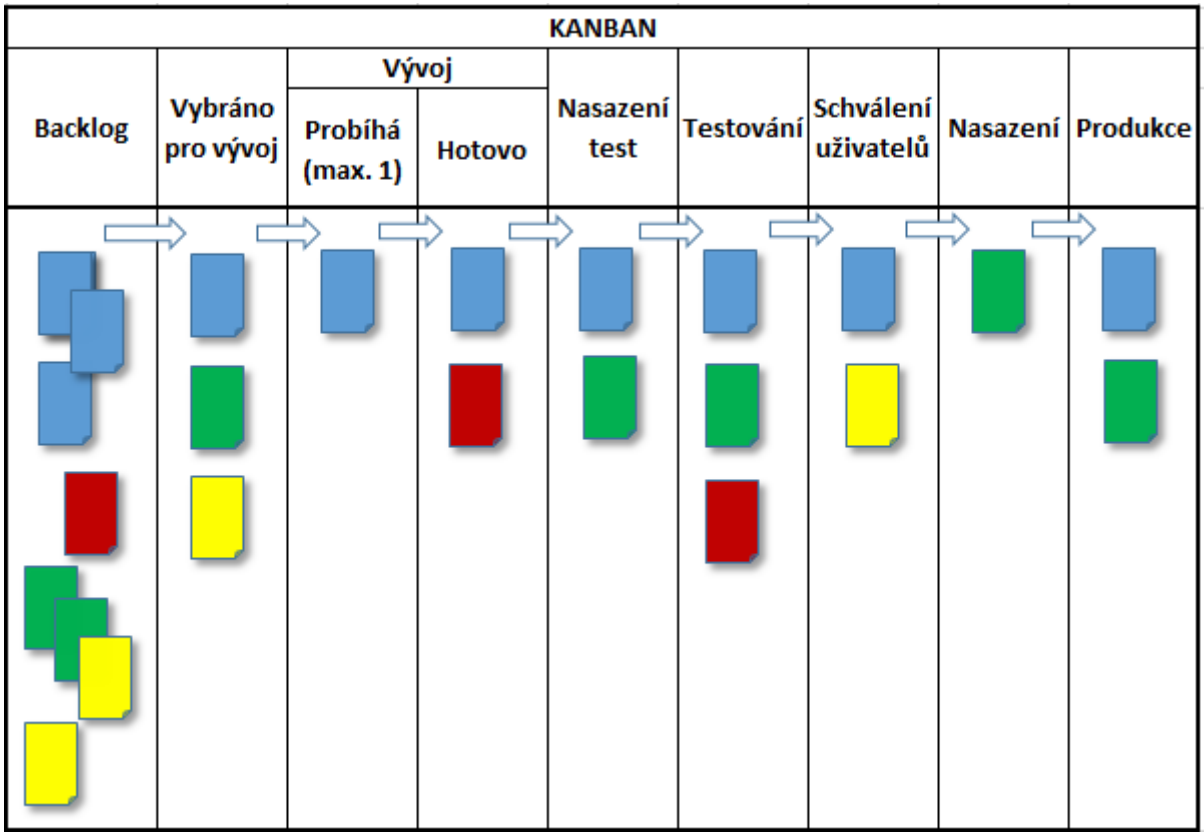
Další základní pravidlo Kanban pro využití přínosu systému tahu je omezení množství rozdělané práce „WIP“ (anglicky work in progress), neboť vždy existují nějaká kapacitní omezení. Toto pravidlo přispívá k tomu, aby se tým soustředil na dokončení rozpracovaných požadavků, protože jinak nemůže začít práce na dalších požadavcích. Pokud dojde k situaci, kdy není možný přechod do dalšího stavu kvůli naplnění omezení rozpracované práce, je to snadno viditelné díky Kanban tabuli. Celý tým se může zaměřit na odstranění tohoto omezení. Zajistí se tak lepší týmová spolupráce i fakt, že jedna část týmu nebude příliš napřed, nebo naopak zaostávat za zbytkem. Nastavení omezení množství rozdělané práce vede ke zkrácení celkového času pro průchod procesem (anglicky Lead time) (Macháčková a Macháček, 2017).

Toto pravidlo vede k tomu, že ve sloupci "Ongoing" mohou být v každém okamžiku nejvýše 2 položky. Tím se liší Kanban tabule od Scrum tabule, kde může být více položek ve sloupci "Ongoing". K přesunutí ze sloupce "Ongoing" na „Done“ je možné, až když je hotový unit test.

Vizuální povaha Kanban umožňuje týmům snadněji komunikovat o tom, jaké práci je potřeba věnovat pozornost, aby zajistily, že tok hodnoty pro zákazníka zůstane v procesu. Kanban je skvělý nástroj pro vizualizaci koncentrace práce, úzkých míst a pomáhá snižovat množství odpadu a maximalizovat efektivitu toku (DevOps Agile Skills Association, 2017, s. 270).

Komplexní Kanban tabule mohou být vytvořeny tak, aby vizualizovaly tok práce přes celou mapu toku hodnot. Na následujícím obrázku je znázorněn příklad komplexnější Kanban tabule.

Obrázek 2.3 – Příklad komplexnější Kanban tabule



Zdroj: vlastní zpracování (Kniberg a Skarin, 2010, s. 42).

2.3 Charakteristika přístupu DevOps

Filozofie DevOps zásadním způsobem přetváří tradiční vztahy mezi vývojem a provozem. Boří bariéry nepochopení mezi různými skupinami, které existují v tradičních organizacích ve formě organizačních nebo procesních hranic a které zabraňují úzké spolupráci. Je to změna způsobu myšlení a kultury, která přináší inovace, rychlost změn a úzkou spolupráci mezi IT a obchodem. Z inovativních přístupů DevOps profitují všechny zainteresované strany – vývoj, provoz, byznys. Jistě je to správný trend, který oprávněně nabírá na síle.

DevOps stojí na agilních principech deklarovaných v Manifestu agilního vývoje softwaru, stojí tak v kontrastu k tradičním metodikám vývoje softwaru jako je vodopádový model nebo metodika RUP.

Kořeny DevOps můžeme nalézt již v roce 2007. Myšlenka na řešení založeném na lepší spolupráci vývoje a provozu vznikla v hlavě Patrika Deboise, který byl rozčarován konflikty mezi vývojáři a systémovými administrátory během práce na migraci datových center pro belgickou vládu (Paul, 2014).

O rok později na konferenci Agile 2008 v kanadském Torontu navrhl Andrew Shafer, softwarový vývojář, diskusní téma pro ad hoc sezení s názvem „Agile Infrastructure“, které však zrušil z nedostatku zájmu. Patrick Debois byl jediný, kdo se chtěl zúčastnit. Oba se následně setkali a vedli rozsáhlou diskuzi, která vedla ke vzniku skupiny Agile System Administration Group. Na návrh Johna Allspaw and Paula Hammonda se rozhodl v říjnu 2009 uspořádat vlastní konferenci s názvem DevOpsDays, které se účastnila celá řada odborníků z IT. S koncem konference se diskuze přesunula na Twitter, kde Debois vytvořil snadno zapamatovatelný hashtag #DevOps. Od té doby nese hnutí označení DevOps, kterého se postupně chytla široká IT komunita a pak i IT organizace (Paul, 2014).

Existuje mnoho různých definic pojmu DevOps. Některé jsou více přesné, jiné méně.

Přístupu DevOps je třeba rozumět jako:

„*Devops je mix vzorců, které mají zlepšit spolupráci mezi vývojem a provozem. DevOps se zaměřuje na sdílené cíle a motivy, stejně jako na sdílené procesy a nástroje. Kvůli přirozeným konfliktům mezi různými skupinami společné cíle a motivy nemusejí být vždy dosažitelné. Měli by však být alespoň navzájem sladěny.*

Devops respektuje skutečnost, že firmy a projekty mají specifické kultury a že lidé jsou důležitější než procesy, které jsou naopak mnohem důležitější než nástroje. Devops akceptuje nevyhnutelnost konfliktů mezi vývojem a provozem.“ (Hüttermann, 2012, s. 8)

DevOps využívají organizace, které dosahují vysoce výkonných IT měřítek:

„*DevOps je kulturní a provozní model, který podporuje spolupráci, aby umožnil vysoce výkonné IT dosáhnout obchodních cílů.*“ (DevOps Agile Skills Association, 2017, s. 20)

DevOps dalo vzniknout hnutí IT profesionálů, kteří chtějí překonat tradiční bariéry mezi vývojem a IT provozem s cílem zefektivnit práci v celém životním cyklu softwaru:

„Odborný výraz "DevOps" obvykle odkazuje na vznikající profesní hnutí, které obhajuje společný pracovní vztah mezi vývojem a IT provozem, což vede k rychlému toku plánované práce (tj. vysokým mírám nasazení) a současně zvyšuje spolehlivost, stabilitu, odolnost a bezpečnosti produkčního prostředí.“ (Kim, 2013, s. 4)

Základní kameny přístupu DevOps tvoří trojimperativ aspektů:

- Lidský aspekt
- Procesní aspekt
- Nástroje

Lidský rozměr DevOps je zásadní. Reprezentován je kulturou, která je založena na otevřené komunikaci mezi kompetentními lidmi, přebírání zodpovědnosti a úsilí dělat věci lépe, rychleji a efektivněji. Bez odpovídající kultury není možné mít v organizaci fungující DevOps, neboť záleží na lidech.

Procesní aspekt je založen na používaných procesech životního cyklu vývoje softwaru, jejichž nedílnou součástí jsou procesy agilního vývoje softwaru. Agilní metodiky tvoří metodickou část DevOps, přičemž agilní postupy je možné libovolně kombinovat za účelem dosažení lepšího výsledku. Procesy musejí být zavedené, popsané a neoddiskutovatelné. Procesní řízení umožňuje

identifikovat oblasti a procesy, které nevytváří hodnotu pro byznys, a proto je možné je zajistit přes externí poskytovatele služeb.

Nástrojů pro DevOps je celá řada. Musejí podporovat přístup DevOps, tj. od vývoje softwaru po jeho provoz, a splňovat kritéria výkonu IT spojené s DevOps. Pokud podnik zná a má popsané procesy DevOps, je schopný si vybrat odpovídající nástroje, které mu budou šité na míru.

DevOps Agile Skills Association (2017, s. 19) zdůrazňuje, že DevOps je o komunikaci a o spolupráci vývojových týmů a odborníků IT provozu na společném cíli. Pomáhá jim lépe chápat role a úkoly, aby organizace mohly iterativně a často nasazovat kvalitní a stabilní produkty a služby s co nejméně defekty a nedostatky. Velké projekty rozděluje DevOps na menší dodávky a více nasazení. Malá nasazení se dají lépe spravovat a snadněji ladit od návrhu přes vývoj a nasazení až po jejich uvedení do provozu a stabilizaci. Zároveň je DevOps na jedné straně hnutí IT praktiků, přičemž nikdo nemá obecnou pravomoc na DevOps a ani ho nevlastní. Na druhé straně je to kultura, jež má za cíl zlepšit, jak lidé spolupracují.

Důvody pro zavedení přístupu DevOps jsou dle DevOps Agile Skills Association (2017, s. 16):

- Zlepšení rychlosti uvádění na trh (anglicky speed to market)
  DevOps umožňuje organizacím přecházet z konceptu na životaschopný produkt v krátkém časovém horizontu. Tím jim umožňuje získat konkurenční výhodu.
- Průběžná integrace a dodávka (anglicky Continuous Integration and delivery)
  Umožňují organizacím častěji nasazovat kód v kratších dodacích cyklech. Průběžné integraci a dodávce bude věnována jedna z následující kapitol.
- Vyšší kvalita, méně poruch a vyšší stabilita
  Přístup a kultura DevOps vedou prokazatelně ke zlepšení systémových vlastností, jako je kvalita, stabilita a poruchovost, ve srovnání s organizacemi, které nezavedli DevOps.
- Inovace a kreativita
  Průběžná integrace, standardizovaná produkční prostředí a automatické nasazení umožňují změřit se více na konceptuální práci a podporují tvořivost a vynalézavost.
- Zvýšená angažovanost pracovníků a spokojenost s prací
  Kultura DevOps podporuje prostředí pro spolupráci a posilování schopností všech členů týmu. To vede ke kvalitnějším výstupům, což se promítá do lepších obchodních výsledků.
- Rozbíjení sil a odstraňování odpadu
  Multidisciplinární dovednosti a efektivní spolupráce a komunikace pomáhá odstraňovat organizační sila. Lean praktiky zaměřující se na odstraňování odpadů a DevOps umožňují týmům plnit své úkoly rychle a efektivně při zachování stability a kvality.
- Snižování zdrojů a nákladů
  Využitím přístupu průběžného dodávání (anglicky Continuous Delivery) a postupů Lean managementu je organizace schopna výrazně snížit náklady a poptávku po zdrojích.

Tyto důvody vedou ke zvýšenému výkonu IT.

2.4 Principy DevOps

Cílem této části je shrnout koncept DevOps z hlediska kultury, principů, procesů a začleněných metodik agilního vývoje.

DevOps představuje změnu v oblasti firemní kultury IT, která se zaměřuje na rychlé dodání softwarových řešení nebo poskytování IT služeb prostřednictvím přijetí agilních a Lean praktik. Lze tedy konstatovat, že základní zásady agilního vývoje a Lean IT jsou jádrem DevOps.

DevOps se orientuje na lidi a kulturu a snaží se zlepšit komunikaci a spolupráci mezi vývojovými a provozními týmy. V DevOps se využívají technologie, a to zejména automatizační nástroje.

Dle DevOps Agile Skills Association (2017) by měly organizace prostřednictvím DevOps, Lean Startup a inovací managementu směřovat k vytvoření prostředí, kde nevadí problémy a jsou vnímány jako příležitosti dělat věci lépe. V takovém prostředí se problémy využívají k vylepšení původního stavu.

Filosofie DevOps je dle Farenhorsta a Loadera (2016) založena na následujících šesti hlavních principech:

- Počínání zaměřit na zákazníka
  Veškeré aktivity při vytváření produktů a služeb se točí kolem zákazníka, jenž je v centru dění. Každý má svého klienta. Každý tým by měl mít odvahu dělat věci samostatně.
- Tvořit s vědomím konečného výsledku
  Všichni členové týmu vědí, jaký bude konečný výsledek. Cokoliv tvoří, mají rozmyšleno, co je tím cílem, kam směřují. To podporuje myšlení zaměřené na produkty a služby a na spolupráci. Spolupráce musí být naprosto férová.
- Celostní odpovědnost
  Celý tým má plnou (anglicky end-to-end) odpovědnost od začátku myšlenky či konceptu až po ukončení produktu nebo služby. Každý den žije zodpovědností za návrh řešení, vývoj, integraci, testování, nasazení, release, údržbu a podporu.
- Multidisciplinární autonomní týmy
  Pro multidisciplinární tým (anglicky cross-functional) platí, že musí mít pokryté všechny potřebné funkce, tedy být složen ze zástupců všech specializací. Zároveň by ale každý měl mít základní znalost toho, co dělají ostatní, aby se tým stal autonomním.
- Neustálé zlepšování
  Všichni členové týmu se zaměřují na neustálé zlepšování s cílem minimalizovat odpad, optimalizovat rychlost, náklady, usnadnit dodávku a zlepšit kvalitu. Zkouší, provádí experimenty a z chyb se učí.
- Automatizovat vše, co můžete
  Automatizace pomáhá zvyšovat kvalitu a maximalizovat tok.

Tyto principy tvoří silné stránky DevOps a jejich použití během digitální transformace zajišťuje nejlepší návratnost investic.

Přínosy DevOps dle (DevOps Agile Skills Association, 2017, s. 13) jsou následující:

- Rychlejší dodání řešení
- Snížení nákladů v dodávce řešení
- Prostor pro inovace
- Zlepšení kontinuity a stability

Výše uvedené lze shrnout tvrzením, že hlavním motivem je dodat kvalitní softwarové řešení rychle a pokud možno levněji.

Kim (Kim et al. 2016, s. 11-13) mluví o DevOps jako výsledku uplatnění Lean principů na tok hodnot IT a uvádí následující soubor tří hlavních principů (The Three ways), na nichž stojí DevOps:

1. Princip umožňuje rychlý tok práce zleva doprava z vývoje do provozu k zákazníkovi, jak je znázorněno na obrázku *2.4 - První princip - tok práce*. Maximalizace a zrychlení toku vyžaduje zviditelnit práci, snížit velikosti celků, zkrátit intervaly, budovat kvalitu prostřednictvím předcházením defektů a neustále optimalizovat pro celkové cíle. Výsledné praktiky zahrnují procesy průběžného sestavení, integrace, testování a nasazování, vytváření prostředí na vyžádání a omezení množství rozdělané práce (WIP).

   Obrázek 2.4 – První princip - tok práce

   Zdroj: vlastní zpracování (Kim et al. 2016, s. 12).

2. Princip umožňuje rychlý a stálý tok zpětné vazby zprava do leva ve všech fázích toku hodnot. K tomu je potřeba zesílit zpětnou vazbu, aby se zabránilo opakování problémů nebo aby se umožnila rychlejší detekce a obnova a tím se předcházelo katastrofickým selháním. Pomocí vidění problémů, když se vyskytnou, jejich shlukováním, dokud nedojde k účinným protiopatřením, je možné neustále zkracovat a rozšiřovat zpětné vazby a maximalizovat schopnost organizace učit se a zlepšovat. Princip zpětné vazby je znázorněn na následujícím obrázku.

   Obrázek 2.5 – Druhý princip - zpětná vazba

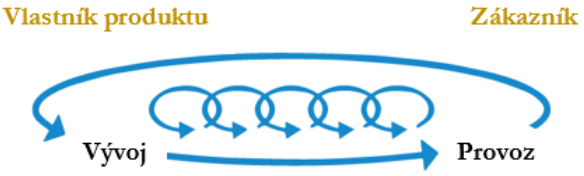
   

   Zdroj: vlastní zpracování (Kim et al. 2016, s. 12).

3. Princip umožňuje vytvořit tvůrčí kulturu vysoké důvěry, která podporuje dynamický, disciplinovaný a vědecký přístup k experimentování a riskování a která usnadňuje učení se z úspěchů a nezdarů. S pomocí zlepšování zpětné vazby (druhý princip) je možné

dosáhnout vytvoření bezpečnějších systémů práce a zlepšení schopnosti riskovat a provádět experimenty, které pomáhají učit se rychleji než konkurence. Součástí tohoto principu je i vytvoření systému práce v organizaci, který umožní znásobit účinky nových znalostí a přeměnit lokální objevy na celková zlepšení. Třetí princip neustálého učení a experimentování je znázorněn na obrázku *2.6 - Třetí princip - neustále učení a experimentování.*

Obrázek 2.6 – Třetí princip - neustále učení a experimentování



Zdroj: vlastní zpracování (Kim et al. 2016, s. 12).

Dle Puppet Labs (2014, s. 18) uvádí následující špičkové prediktory[3], které vedou vyšší výkonnosti IT v DevOps prostředí:

- Proces schvalování změn založený na vzájemném hodnocení
  O změnách je nutné nechat rozhodovat ty, kdo je dělají, a tým učinit odpovědným za kvalitu změn prostřednictvím vzájemného hodnocení. Rozhodnutí někoho mimo tým nemá váhu. Tím se zrychlí proces schvalování změn.
- Řízení verzí pro všechny produktivní artefakty
  Pro každý artefakt je nutné mít jednu verzi pravdy, tj. jeden zdroj dat. V případě problémů je možné s pomocí dobrého řízení verzí vrátit k poslední funkční verzi a zkrátit tak dobu potřebnou k obnovení.
- Proaktivní monitoring produktivního prostředí
  Týmy jsou schopny řešit problémy rychleji díky proaktivnímu sledování vlastních produktů a služeb a tím urychlovat jejich neustálé zlepšování.
- Organizační kultura založená na vysoké důvěře
  Pokud pracovníci mají vysokou důvěru, nemusí se některými věcmi zabývat. Vysoká důvěra zrychluje jednání, a snižuje náklady s ním spojené.
- Vztah mezi vývojem a provozem musí být úspěšný pro obě strany (anglicky Win-win)
  Není možné dělat nějakou věc na úkor někoho jiného, například na úkor lidí z provozu.

K těmto výše uvedeným prediktorům výkonnosti IT přidává Kim Gene (Paul, 2015) další dva prediktory výkonnosti IT:

- Průběžná integrace a nasazení
- Automatizované akceptační testy

Průběžné dodávání (anglicky Continuous delivery) a organizační kultura mají největší vliv na výkonnost IT! (Puppet Labs, 2016)

[3] Pojem „Prediktory výkonnosti IT“ je uveden v terminologickém slovníku, viz kapitola *Terminologický slovník*.

2.5 Holistický přístup DevOps od vývoje po provoz softwaru

Cílem této části je poskytnout komplexní pohled na celý proces dodávání softwaru a IT provozu v kontextu holistického přístupu DevOps, který všechna témata spojuje do soudržného celku. DevOps je totiž potřeba posuzovat na základě dalších přístupů, které byly vyvinuty a rozšířeny nezávisle. Nicméně DevOps je přejímá, integruje a tím se stávají vlastnostmi DevOps prostředí. Aby bylo možné zabývat se DevOps, je nutné zkoumat ho z hlediska všech jeho vlastností.

DevOps prostředí se vedle agilního vývoje software vyznačuje následujícími přístupy, které mají své kořeny v špičkových výrobních metodách jako je Kanban a Lean, a které jsou dále vysvětleny:

- Průběžná integrace (anglicky Continuous Integration - CI)
- Průběžné testování (anglicky Continuous Testing)
- Průběžné dodávání (anglicky Continuous Delivery – CDE)
- Plynulé nasazování (anglicky Continuous Deployment - CD)
- Nepřetržitý monitoring a zpětná vazba (anglicky Continuous Monitoring and Feedback)
- Neustálé zlepšování (anglicky Continuous improvement)
- Infrastruktura jako kód (anglicky Infrastructure as Code)
- Automatizace (anglicky Automation)

Přístupy Continuous Integration, Continuous Delivery a Continuous Deployment, které se zpravidla používají v anglické podobě, byly v této práci přeloženy do českého jazyka. Přitom anglické slovíčko „Continuous“ bylo přeloženo vždy v nejvhodnější formě v kontextu daného přístupu. Například pokud vývojáři provádějí průběžně integraci kódu, ale ne výslovně nepřetržitě, je Continuous Integration přeložen jako Průběžná integrace. Stejným způsobem je přeložen Continuous Delivery jako Průběžné dodávání. Pokud je dodávka a nasazení balíčku plně automatizovaná až do produkce, tedy tok kódu probíhá plynule bez ručního zásahu, jako je tomu u Continuous Deployment, je tento přístup přeložen jako Plynulé nasazování.

Průběžná integrace (anglicky Continuous Integration - CI)

Průběžná integrace je vývojová praktika, která je neodmyslitelnou součástí agilních kultur a tím i DevOps přístupu. Postavena je na řízení verzí, které má tu schopnost, že umožňuje větvení kódu. Větve kódu (anglicky branches) umožňují vývojářům pracovat samostatně každý na své části kódu a nezávisle na hlavním kmenu (anglicky trunk) a ostatních větvích. Nicméně, aby to takto mohlo fungovat, je nutné po dokončení změn přenést kód z větví zpět do hlavního kmene sloučením s ostatními změnami pomocí operace merge, integrovat kód, sestavit (build) a otestovat ve vývojovém prostředí. Pokud vzniknou konflikty, je třeba je řešit. Do kmene se nesmí zavléct chyby nebo ho jinak destabilizovat. Zde přichází ke slovu průběžné integrace, kterou vystihují následující definice:

„Průběžná integrace je praktika ve vývoji softwaru, kdy členové týmu často integrují svou práci, obvykle každý člověk integruje alespoň denně - což vede k více integracím za den. Každá integrace je ověřena automatizovaným buildem (včetně testování), ve kterém se co nejrychleji detekují

integrační chyby. Mnoho týmů zjistilo, že tento přístup vede k výrazně sníženým integračním problémům a umožňuje týmu rychleji vyvíjet kompaktní software.“ (Fowler, 2006)

„Všichni vývojáři provozují soukromé buildy na svých vlastních pracovních stanicích předtím, než provedou commit svého kódu do repositáře pro řízení verzí, aby zajistili, že jejich změny nenaruší integrační build.“ (Duvall, 2007)

Jak je vidět z definice průběžné integrace, je založená na každodenní průběžné a časté integraci práce vývojářů do jednoho celku při zachování integrity (funkčnosti). Časté integrace a menší balíčky předcházejí složitým konfliktům kódu.

Největší předností průběžné integrace je automatizace. Kompilace se spouští automaticky, stejně tak i sestavení na integračním stroji po každém příkazu commit. Rovněž ověření provedených změn probíhá ve vývojovém prostředí na softwaru jako celku automatickými Unit testy, které pomáhají dříve identifikovat případné nedostatky a chyby. Testování přitom nelze vynechat. Automaticky se může spustit i přemístění na jakýkoliv server (repositář). Další výhodou je hlídání nastavených pravidel psaní kódu, které si sami vývojáři definují pro svou práci. Díky každodennímu provádění příkazů commit vývojáři do hlavní větve a automatickému sestavení (buildu) je snadno dostupná poslední spustitelná verze softwaru. Průběžná integrace je nezbytným předpokladem pro automatizované Průběžné dodávání (Continuous Delivery).

Průběžné testování (anglicky Continuous Testing)

Průběžné testování znamená, že se testuje dříve a nepřetržitě napříč celým životním cyklem softwaru. Na ověřování kvality testováním se podílí jak vývojáři, tak i provoz. Vývojáři testují svůj kód, podílejí se na integračních testech, doporučují testovací metody a pomáhají definovat testovací prostředí. Provoz spolupracuje na monitorování testovacího prostředí, stará se o jeho konfiguraci a podílí se na testech. V tomto kontextu přináší průběžné testování a automatizace úspory času, zvýšení kvality a snížení nákladů potřebných na testování.

Průběžné dodávání (anglicky Continuous Delivery - CDE)

Koncept průběžného dodávání navazuje na průběžnou integraci. Je třeba rozlišovat, že průběžné dodávání není synonymum pro DevOps. Jen mají oba koncepty společné základy.

Průběžné dodávání umožňuje série postupů, jejichž cílem je zajistit, že kód může být rychle a bezpečně nasazen do produkčního prostředí, a to prostřednictvím častého automatizovaného nasazování změn do předprodukčního prostředí. Stejně tak jsou pomocí automatizace prováděny testy (Slatinský, 2016, s. 13).

Humble a Farley (2010) předkládají následující definici: *"Průběžné dodávání je o předání release plánu do rukou byznysu, ne do rukou IT. Implementace průběžného dodávání znamená zajistit, že váš software je vždy připraven do produkce po celý jeho životní cyklus – že jakýkoli build by mohl být uvolněn uživatelům stisknutím tlačítka s pomocí plně automatizovaného procesu během několika sekund nebo minut."*

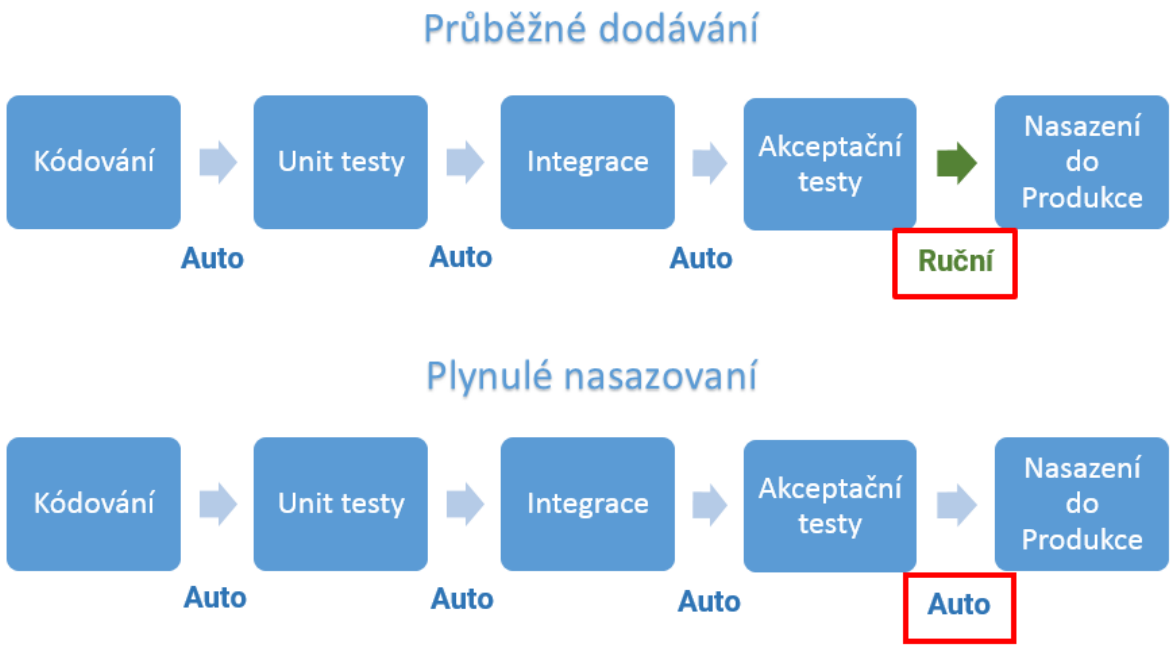
Mezi klíčové zásady průběžného dodávání patří rigorózní automatizace, extrémní zpětná vazba a průběžné změny. Pro týmy, které přijaly průběžné dodávání, přináší tento přístup zvýšení rychlosti a opakovatelnosti pomocí automatizace, jsou převážně autonomní a zcela agilní, protože

neexistuje žádná rozdělaná práce, podporují tok ve své dodávce a dělají věci správným způsobem (DevOps Agile Skills Association, 2017, s. 266).

Plynulé nasazování (anglicky Continuous Deployment - CD)

Plynulé nasazování spočívá v průběžném automatizovaném nasazování změn přímo do produkčního prostředí. Charakteristické pro plynulé nasazování jsou časté releasy, které procházejí sérií automatizovaných testů. Rozdíl mezi průběžným dodáváním a plynulým nasazování spočívá v tom, že průběžné dodávání má ruční release k nasazení do produkce (po úspěšných akceptačních testech), zatímco plynulé nasazování má releasy automaticky pouštěné do produkce. V průběžném dodávání je release rozhodnutí lidí (DevOps Agile Skills Association, 2017, s. 154). Rozdíl mezi oběma koncepty je znázorněn na následujícím obrázku.

Obrázek 2.7 – Rozdíl mezi průběžným dodáváním a plynulým nasazováním



Zdroj: vlastní zpracování (DevOps Agile Skills Association, 2017).

Nepřetržitý monitoring a zpětná vazba (anglicky Continuous Monitoring and Feedback)

Nepřetržitý monitoring je klíčem k tomu zjistit, co se děje. Nepřetržitým monitoringem a zpětnou vazbou od zákazníka je sníženo riziko spojené s častějším nasazováním releasů a většího množství kódu do produkčního prostředí. V rámci nepřetržitého monitoringu jsou sledovány infrastruktura, servery (tzv. server monitoring) a výkon aplikací. Cílem je měřením posílit schopnosti identifikace, vyhodnocení a iniciaci řešení případného výpadku nebo chyby v průběhu celého životního cyklu aplikace (Bestpractice.cz, 2014).

Moderní technologie umožňují sbírat informace o chování zákazníka, mimo jiné přímo během používání aplikace. To pomáhá identifikovat chyby. Tyto informace jsou cenným zdrojem při samotných řešeních, zlepšovaní aplikace a zákaznické spokojenosti. Tato forma zpětné vazby

umožňuje organizaci být neustále více pružnou a schopnou reagovat na potřeby zákazníka (Bestpractice.cz, 2014).

Neustálé zlepšování (anglicky Continuous improvement)

Neustálé zlepšování je neodmyslitelnou součástí kultury DevOps. Na neustálé zlepšování je kladen velký důraz, aby se zlepšily produkty či služby nabízené zákazníkům. Stěžejní myšlenkou je, že všechno se dá dělat lépe. Cíli neustálého zlepšování je dodávat hodnotu rychleji, lépe a levněji. Zároveň i vytvořit více smyslu v práci a zdravější životní prostředí (DevOps Agile Skills Association, 2017, s. 154).

Neustálé zlepšování je přístupem z Lean prostředí k identifikaci příležitostí pro zjednodušení práce a snížení odpadu. Zaměřuje na přizpůsobování a učení se z neúspěchu prostřednictvím strukturovaného řešení problémů. Týmy se musí soustředit na experimentování, rychlá selhání, minimalizaci plýtvání, optimalizaci rychlosti, nákladů a hladké dodávky, aby mohly neustále inovovat a provádět potřebné změny (DevOps Agile Skills Association, 2017, s. 27). Problém je potřeba vnímat jako příležitost dělat věci lépe. Problémy jsou vítané.

Platí zásada, že co nelze měřit, nelze zlepšit. Bez měření není možné dosáhnout a udržet neustálé zlepšování. DevOps se snaží měřit procesy, lidi a nástroje pomocí metrik výkonu, procesů, inovací, kultury a podpory (DevOps Agile Skills Association, 2017, s. 27). Organizace často používají pro zabezpečení neustálého zdokonalování iterativní Demingův cyklus PDCA (anglicky plan-do-check-act tedy naplánuj-proveď-ověř-jednej).

Lze konstatovat, že pro tento přístup je třeba, aby ho měl každý v krvi, není možné, abych na to byl v týmu nějaký specialista.

Infrastruktura jako kód (Infrastructure as Code)

Infrastruktura jako kód je nástrojem, který umožňuje organizaci optimálně řídit vytváření a konfiguraci prostředí pro plynulé nasazování. Díky novým technologiím pro virtualizaci je možné na infrastrukturu a konfiguraci systémů pohlížet jako na zdrojový kód, ve smyslu softwarového projektu (Slatinský, 2016, s. 13). Předpokladem je, že infrastruktura je navržena, implementována a nasazována stejným způsobem.

Pokud definuje organizace standardní konfigurace serveru (parametry), může vývojář napsat program pro instalaci a konfiguraci virtuálního serveru, který vyloučí ruční práci. V případě potřeby se klikne na tlačítko, které spustí program, a virtuální server je automaticky vytvořen a nakonfigurován. Stejně tak je možné softwarově nadefinovat prostředí.

Automatizace (Automation)

Automatizace procesů integrace, sestavování, testování a nasazování přispívá k zajištění kvality, rychlosti a nasazování velkého množství měnícího se kódu díky vyloučení lidských chyb, kterého mohou nastat při ručním provádění těchto kroků. Automatizace je možná jen s neoddělitelnou podporou vhodných automatizačních nástrojů. Mezi výhody automatizovaných postupů patří časová úspora a včasná detekce vad softwaru (defektů).

2.6 Týmová organizační struktura pro DevOps

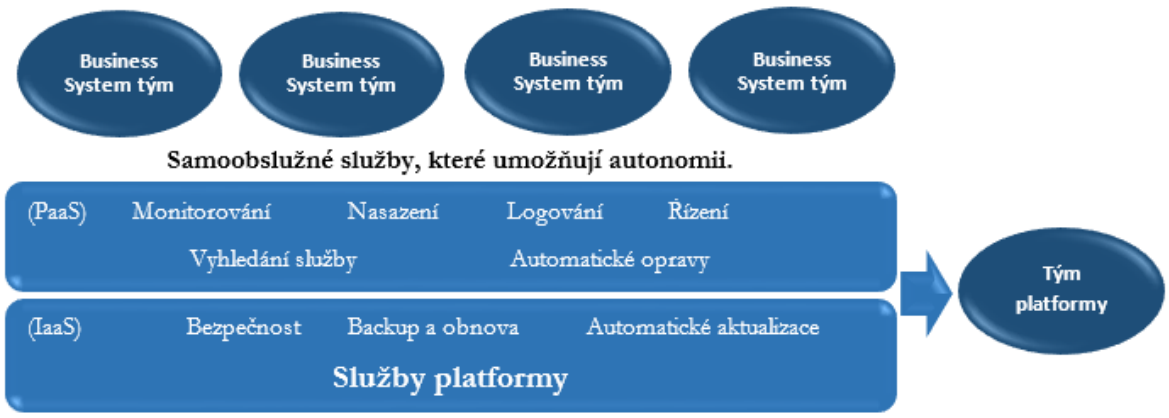
Tato část práce se zabývá přístupy k organizování týmů v prostředí DevOps. Tato kapitola slouží k naplnění dílčího cíle **DC2**, jehož součástí je analýza a popis organizační struktury, na níž je založen holistický přístup DevOps.

Dle DevOps Agile Skills Association (2017, s. 81) je současným preferovaným způsobem organizace týmů DevOps rozčlenění na nezávislé Business System týmy a jeden nebo vícero týmů Platformy. Jedná se praktický přístup. Business System týmy jsou autonomními týmy, které se specializují v dodání specifických softwarových produktů pro obchodní oddělení organizace. Mimoto spravují koncové uživatele, zákazníky, služby a softwarové produkty jako například aplikace a zároveň zaručují jejich kvalitu zákazníkovi. Na druhou stranu týmy Platforem se zaměřují na rozvoj, údržbu a provoz platform, na které Business System týmy umisťují své aplikace a kód infrastruktury. Týmy Platformy tedy řídí platformové produkty používané Business System týmy. Nepřebírají odpovědnost za služby nabízené Business System týmy. Nabízejí platformu (nebo infrastrukturu) prostřednictvím samoobslužných služeb a automatizace. Díky tomu a společné platformě fungují Business System týmy efektivněji. Nemusejí čekat na to, než jejich žádosti budou řešeny týmem Platformy. Oba týmy společně dodávají zákazníkovi ICT služby, které může využívat ve svých podnikových procesech. Proto je důležitá úzká spolupráce mezi nimi, aby zajistily, že služby poskytované týmem Platformy jsou správné. Infrastruktura je v tomto organizačním uspořádání využívána nejvyšší možnou měrou.

DevOps Agile Skills Association (2017, s. 80) varuje před tím, aby si organizace ponechávali stávající organizační strukturu s centralizovanými funkcemi v podobě pevných týmů a odborníků na middleware, databáze, sítě a další a začleňovali do ní nově vznikající Business Systém týmy, neboť to vede k mnoha závislostem a k "plýtvání", které znemožňuje různým Business System týmům pracovat zcela nezávisle na sobě.

Hlavní motivací pro vytvoření autonomních týmů v rámci DevOps je snížení množství odpadu při vývoji a dodávce produktů a zaměření na hodnotu pro koncového uživatele. Každý Business System tým přebírá plnou zodpovědnost za své služby v průběhu celého jejich životního cyklu. Na následujícím obrázku jsou znázorněny autonomní týmy v prostředí DevOps.

Obrázek 2.8 – Autonomie týmů v prostředí DevOps



Zdroj: vlastní zpracování a překlad dle (DevOps Agile Skills Association, 2017).

Business System týmy jsou produktově zaměřené, neboť v prostředí DevOps se jedná o rychlost a získání funkcí pro zákazníka co nejdříve. Rychlé dodávání produktů pomáhá organizacím využívat zpětné smyčky, aby zjistily, že jsou na správné cestě.

Pro úspěch tohoto organizačního modelu DevOps je nutné, aby ho přijala celá organizace a přizpůsobila se mu.

Přijetí principu průběžného dodávání (CD) v rámci organizace vyžaduje, aby byly týmy skutečně nezávislé. Nezávislé týmy mají následující vlastnosti dle DevOps Agile Skills Association (2017, s. 84-85):

- Týmy mají plnou odpovědnost. Neexistuje žádné předání nebo transfer odpovědnosti.
- Týmy pracují převážně autonomně a převážně navzájem nezávisle, aby poskytovaly nepřetržitý tok změn.
- Technická rozhraní mezi Business System týmy a týmy Platformy jsou jasně definovány pomocí aplikačních programovacích rozhraní (API).
- Týmy Platformy poskytují širokou sadu standardizovaných samoobslužných funkcí Business Systém týmům, jako je logování, monitorování, nasazování, zálohování, obnovení a mnoho dalších. Tyto funkce a služby umožňují Business System týmům plně spravovat své softwarové produkty.
- Business System týmy jsou zákazníky týmů Platformy. Používají produkty týmů Platformy, které lze použít jako plně automatizované samoobslužné služby. Tyto služby nezávisí na žádném systému žádostí a ruční vyplňování žádostí je vyloučeno.
- Týmy Platforem jsou odpovědné za vlastnosti platformových produktů, jako je dostupnost a výkon. Business System týmy jsou odpovědné za vlastnosti svých produktů (aplikací), jako je dostupnost a výkon.
- Systémové vlastnosti platformních produktů mohou být monitorovány a řízeny nezávisle na dostupnosti aplikačních produktů, které používají platformové produkty.

3 Outsourcing vývoje softwaru

Tato kapitola je zaměřena na vysvětlení pojmu outsourcing IS/ICT a vzhledem k cílům práce se podrobněji zabývá aspekty specifickými pro jednu z forem částečného outsourcingu IS/ICT a to vývoje softwaru. Obsah této kapitoly přímo slouží k dosažení dílčího cíle práce **DC3**.

Lze konstatovat, že v oblasti IS/ICT roste každoročně poptávka po zajišťování činností externími dodavateli a outsourcing IS/ICT získává na významu mimo jiné v souvislosti s digitalizací mnoha oborů, do kterých proniká. Pro mnoho technologických podniků se stal jednou z klíčových oblastí podnikání. K tomu lze dodat, že společnosti, které se specializovaly, jsou často vysoce výkonné a nabízí zákazníkům vyšší přidanou hodnotu, než jaké by dosáhly vlastními silami (Insourcing).

Pojem Outsourcing se v češtině ustálil jako převzatý výraz z anglického jazyka a znamená převedení určitých funkčních oblastí, které nesouvisí s hlavním předmětem podnikání, na externí poskytovatele služeb, kteří za úplatu za ně převezmou odpovědnost a zajistí jejich vykonávání formou poskytnuté služby. Rozhodnutí o outsourcingu má strategický význam a umožňuje podniku zaměřit se na řízení hlavních činností, které mu přinášejí přidanou hodnotu. K rozhodnutí o převedení ICT služby na externí poskytovatele dochází z pravidla tehdy, jsou náklady na interní zajištění služby vyšší, než je cena, kterou nabídne externí poskytovatel. Kromě toho, že umožňuje snížení nákladů, přispívá k lepším reakcím na potřeby zákazníka, urychlení uvedení produktů na trh, posílení postavení na trhu a k využívání aktiv poskytovatelů a jejich odborných znalostí. Svým charakterem je outsourcing také nástroj změny organizační struktury, který může zajistit efektivnější využití podnikových zdrojů. Jednou z důležitých oblastí, ve které může být uplatněn outsourcing, je oblast IS/ICT.

3.1 Charakteristika outsourcingu IS/ICT

Outsourcing IS/ICT je v tomto kontextu využívání externích poskytovatelů služeb za účelem efektivní IT podpory podnikovým procesům a jejich výsledkům. Podnikům umožňuje podle potřeby externě si zajistit a využívat ICT služby, ICT procesy a ICT zdroje, které podnik nemá nebo je má, ale drahé nebo na nedostatečné úrovni. Dále pomáhá řešit i otázky technologické infrastruktury, neboť zahrnuje outsourcing formou Cloud Computingu jako jsou modely: softwarové služby (SaaS), infrastruktury jako služby (IaaS) a platformy jako služby (PaaS) nebo také řešení Mobile Cloud. V některých případech předchází outsourcingu převzetí vlastních podnikových IT zdrojů do správy externího poskytovatele služeb.

Pro úplnost je třeba uvést, jaké jsou vazby mezi službami, procesy a zdroji. Aby mohly ICT procesy fungovat, vyžadují ICT zdroje. ICT zdroje vstupují do procesů a jsou v nich spotřebovávány. Účelem procesů je transformace vstupů na výstupy. U ICT procesů se jsou výstupem ICT služby, které jsou určeny pro efektivní podporu podnikových procesů.

V následujících podkapitolách jsou na jedné straně uvedeny existující formy outsourcingu IS/ICT podle definice společnosti Gartner (2005) a podle Brucknera (Bruckner et al. 2012, s. 91) a na straně druhé důležité aspekty outsourcingu IT, mezi které patří proces výběrového řízení pro určení externího dodavatele a smluvní vztahy mezi objednavatelem a externími subjekty, ve kterých se definují pravidla spolupráce a závazky smluvních stran.

V souvislosti s outsourcingem vývoje softwaru jsou v této práci uvedeny i kritické faktory externí dodávky, kterým je věnována samostatná kapitola *3.2.1 Kritické faktory externí dodávky*, a jeho rizika, kterým je věnována kapitola *3.2.2 Rizika spojená s outsourcingem vývoje softwaru.*

Podrobné rozebrání outsourcingu IS/ICT jednotlivých forem je mimo rozsah této práce.

3.1.1 Formy outsourcingu IS/ICT

Outsourcing v oblasti IS/ICT má různé důvody a tím i různé podoby a je označován různými přívlastky. Zahrnuje různé kombinace dohod o partnerství, doby jejich trvání, platebních mechanismů a poskytovaných služeb, procesů, zdrojů a i produktů.

Outsourcing IS/ICT člení jednotlivé poradenské instituce (Gartner, Outsourcing Institute, CIO), metodiky (MMDIS) nebo odborná literatura (Bruckner et al. 2012) mírně odlišně, ale v principu charakterizují totéž. Pro tuto práci jsem vybral členění outsourcingu IS/ICT podle Gartner, kde je patrný určitý posun ke komplexním řešením této problematiky, a podle odborné literatury.

Gartner ve svém výzkumu z roku 2005 (Gartner, 2005) a ve svém volně přístupném IT glosáři[4] na svém webu vymezuje outsourcing jeho účelem, a to jako smluvní vztah s externí firmou za účelem přenesení odpovědnosti za určitou část funkční oblasti. Outsourcing člení do následujících forem:

- Outsourcing infrastruktury
    - outsourcing sítí (anglicky Network Outsourcing)
    - outsourcing osobních počítačů (anglicky Desktop Outsourcing)
    - outsourcing datového centra (anglicky Data Center Outsourcing)
- Outsourcing aplikací (anglicky Applications Outsourcing - AO)
    - outsourcing podnikové aplikace (anglicky Enterprise Application Outsourcing)
- Outsourcing podnikového procesu (anglicky Business Process Outsourcing - BPO)

Zde stojí za zmínku, že konzultanti z Gartner (Gartner, 2018) vyvinuli cyklus zajišťování zdrojů (anglicky Gartner's Sourcing Cycle) o čtyřech fázích, které odpovídají rychlosti a požadavkům podniku. Jeho účelem je řešit nedostatečnou organizaci zajišťování zdrojů ve středních podnicích prostřednictvím vytvoření udržitelné strategie pro získávání zdrojů.

V charakteristice cyklu zajišťování zdrojů již neuvádějí explicitně pojem outsourcing, ale hovoří o zajišťování zdrojů prostřednictvím služeb infrastruktury (anglicky Infrastructure Services), aplikačních služeb (anglicky Application Services) a služeb podnikových procesů (anglicky Business Process Services), které jsou postaveny na cloudových službách (Cloud services).

Z hlediska ICT oblasti organizace lze vytěsnit na externího poskytovatele podnikový proces, ICT službu, ICT proces nebo ICT zdroj (Bruckner et al. 2012, s. 91). Pokud se vezme v úvahu i možné dělení na částečný outsourcing, lze definovat následující základní formy outsourcingu dle předmětu a rozsahu (Bruckner et al. 2012, s. 91):

- Outsourcing podnikového procesu (anglicky Business Process Outsourcing - BPO),
- Outsourcing komplexního IS/ICT,

[4] https://www.gartner.com/it-glossary

- Částečný IS/ICT outsourcing v různých formách
  - aplikační ICT služby (ASP, SaaS, DaaS)
  - ICT procesu
  - ICT zdroje a jeho údržby (PaaS, IaaS).

Pod outsourcingem podnikového procesu se rozumí vytěsnění celého podnikového procesu včetně ICT zdrojů na jeho podporu na externího poskytovatele. Nejčastěji se tato forma outsourcingu uplatňuje u podpůrných procesů.

Outsourcing komplexního IS/ICT znamená vytěsnění vývoje a provozu informačního systému podniku na externího poskytovatele, při kterém jsou převedeny všechny související ICT služby, procesy a zdroje včetně jejich integrace a customizace. Přesun ICT zdrojů zahrnuje software, hardware a pracovníky. Při provozu IS/ICT musí externí poskytovatel dodržovat dohodnuté SLA.

Částečný IS/ICT outsourcing zahrnuje vhodné selektivně vybrané ICT služby, procesy nebo zdroje, které se převedou na externího dodavatele.

Z hlediska ICT služeb sem patří poskytování aplikačních služeb (Application Services Providing – ASP) a outsourcing formou Cloud computingu, typicky se jedná o služby SaaS (Software as a Service) či DaaS (Desktop as a Service). ASP je model pronájmu aplikací přes internet, kdy poskytovatel (Application Service Provider) vzdáleně nabízí a provozuje stejnou, zpravidla velmi málo customizovanou aplikaci pro více zákazníků zároveň. Pro oba modely ASP a SaaS platí, že uživatel má k dispozici vždy aktuální verzi aplikace s posledními změnami. V praxi se však prosazuje distribuční model SaaS. O modelu ASP se v současné době prakticky nemluví.

Z hlediska ICT procesů se typicky například jedná o outsourcing vývoje softwaru (IS/ICT), provozu a údržby aplikací (IS/ICT) běžící na infrastruktuře zákazníka, implementace, integrace IS/ICT a Service desku.

Pro ICT zdroje a jejich údržbu platí, že je možné outsourcingem zajistit služby datového centra, sítí LAN/WAN, koncových stanic a v neposlední řadě i najmout odborníky v oblasti informačních technologií. V případě outsourcing formou Cloud computingu se jedná o služby PaaS (Platform as a Service) a IaaS (Infrastructure as a service).

3.1.2 Výběrové řízení

Pro určení dodavatele musí existovat standardizovaný postup ve formě výběrového řízení jako přijímacího procesu. Vyhraje ten, kdo lépe splní stanovené podmínky. S dodavateli komunikují určení členové výběrové komise.

Řízení na výběr dodavatele má zpravidla několik kroků. Na začátku je nutné si rozmyslet, čeho chci dosáhnout, jaké mám potřeby a podle toho stanovit základní požadavky a kritéria na dodavatele, které musejí splnit. V konečném výsledku musejí být ověřitelné z hlediska hodnocení úspěšnosti či neúspěšnosti projektu.

V dalším kroku je nutné vytipovat IT firmy na trhu služeb IS/ICT, které se specializují na vývoj softwaru. Po zúžení výběru je možné je oslavit s žádostí o dodatečné informace (anglicky Request for information – RFI) za účelem získání dalších informací a zjištění, zda mají zájem. V případě

častého externího vývoje softwaru mohou být IT firmy předem vybrané na základě dobrých zkušeností nebo referencí a ty jsou pak oslovovány přednostně.

V případě potvrzení zájmu dodavatelů se rozesílá zadání s konkrétními požadavky a kritérii na vývoj softwaru ve formě žádosti o nabídku (anglicky Request for proposal - RFP).

Oslovené softwarové společnosti posílají své nabídky do stanoveného termínu. Zpravidla většina z nich přijde a odprezentuje své návrhy. Výběrová komise je zhodnotí nejprve po formální stránce, pak je vyhodnotí po obsahové stránce a určí ty, kdo nejlépe splňují požadavky a hodnotící kritéria, které byly stanovené na počátku. Hodnotí se rovněž faktory, jako jsou reference, finanční stabilita a důvěryhodnost dodavatelů, aby se zvážilo riziko případného úpadku dodavatele. Pokud nejsou splněné nějaké povinné podmínky, jsou nevyhovující nabídky vyřazeny. Na závěr hodnocení se rozhodne o vítězném dodavateli.

V posledním kroku se pak podle výsledku rozhodnutí vyjednává a podepisuje smlouva s externím dodavatelem.

Kromě toho je možné vyhlásit separátní výběrové řízení na servisní smlouvu. Po vybrání vítězného dodavatele, který bude zajišťovat provoz aplikací, se vyjednává a podepisuje servisní smlouva a dohoda o poskytovaných služeb (Service Level Agreement - SLA).

V případě potřeby odborníka nějaké profese, od analýzy, přes vývoj až po projektové řízení, je možné využít tzv. bodyshop, tzn. nájem přes agenturu nebo služeb softwarových společností, které také nabízejí zkušené odborníky. V druhém případě je výhodou, že mají za sebou ještě síť kolegů, kteří jim umí pomoci a okamžitě je doplnit.

Proces řízení na výběr externisty či bodyshopu je podobný procesu na výběr externího dodavatele. Jen s tím rozdílem, že specializovaná personální agentura či softwarová společnost je oslovena objednávkou.

V případě bodyshopu jednotlivec, externista přijde normálně na pracovní pohovor. Předpokladem by mělo být, že je před procesem RFI odhadnuta a potvrzena pracovní zátěž. Externisti přes bodyshop pracují v kanceláři s ostatními zaměstnanci a vyplňují výkaz vykonané práce (tzv. timesheet), který následně slouží jako podklad pro fakturaci. Odpovědnost za pracovní výstupy externisty přes bodyshop mají příslušní manažeři ať už projektu nebo útvaru. Je nutné zajistit, aby předání know-how bylo zahrnuto do rozsahu práce externistů.

Naopak externista poskytovaný softwarovou společností, který pracuje na základě servisní smlouvy nebo smlouvy o dodávce, nesedí fyzicky u zákazníka a ten si ani nevybírá konkrétního člověka. Vůbec ho nevidí. Svou práci pro zákazníka vykazuje své IT firmě. Jako externista se servisní smlouvou mohou být najati lidé různých profesí - aplikační správce i vývojář.

3.1.3 Smluvní vztahy

V této kapitole se zabývám vztahy mezi objednavatelem a třetími stranami, kterými mohou být externí dodavatelé softwaru, provozovatelé softwaru a poskytovatelé služeb.

Smluvní vztah musí mít pravidla. Smlouva je závazkový dokument obsahující podmínky spolupráce. Smluvní vztahy se v České republice v obecné rovině řídí novým občanským zákoníkem.

Objednavatel vstupuje do smluvních vztahů s externími partnery za účelem zajištění právní jistoty a s cílem vyhnout se případným nedorozuměním, konfliktům, a sporům pro případ, že si obě strany budou rozdílně vykládat představy ohledně výsledného díla a procesů tvorby a implementace softwaru. Prokázané porušení vzájemné dohody musí být však spojené s vyvozením příslušných následků, například ve formě penále. Pro tyto účely je podstatné vymezení a upřesnění předmětu smlouvy a správné formulace článků. Je třeba se vyvarovat příliš obecných a vágních formulací. V českém právním prostředí se v rámci softwarového práva používají následující typy smluv (Jansa et al. 2018):

- Smlouva o dílo
  Základním smluvním typem, na základě kterého se zpravidla vyvíjí a implementuje software, je Smlouva o dílo[5] s licenční smlouvou. Týká se na jedné straně zhotovení softwaru na objednávku, ale i implementace, tj. způsobu, jak je objednavateli dodán a uveden do provozu. Smlouva o dílo[6] musí standardně obsahovat podstatné náležitosti a ustanovení týkající se vymezení předmětu díla, doby a způsobu jeho vytváření, splatnosti ceny za dílo či ochrany důvěrných informací. Kromě těchto ustanovení se ve smlouvě o dílo řeší otázky specifické pro tuto oblast.
- Rámcová smlouva
  Je třeba si uvědomit, že v agilním světě je zákazník partnerem. To znamená, že není vyloženě jen smluvní stranou, která pošle na začátku specifikaci zadání a následně očekává výsledek podle této specifikace. Tomu je nutné přizpůsobit i typ smluvního vztahu. V praxi se pro agilní vývoj nejčastěji uplatňuje Rámcová smlouva, která určuje základní rámec jednotlivých smluvních vztahů dle potřeb objednavatele s tím, že obsah dílčích smluv bude dán dohodou smluvních stran v průběhu realizace softwarového projektu. Rámcová smlouva upravuje především proces vzniku dílčí smluv na základě objednávek. Zahrnuje cenotvorbu pro dílčí smlouvy, řešení problematiky duševního vlastnictví, platební podmínky, odpovědnosti za vady a za škodu a podmínky ukončení smluv. Dílčí smlouvy mohou být sjednány pro analýzu, programátorské práce, dodávku hardwaru, dodávku softwaru a implementační práce.
- Smlouva o analýze v IT
  Lze ji samostatně sjednat pro podrobnější analýzu pro zjištění požadavků a potřeb objednavatele, anebo pro zjištění stavu IT infrastruktury či úrovně využití dosavadního softwaru. Uplatňuje se všude tam, kde je potřeba provést předimplementační analýzu, tj. u dodávek softwaru na míru, customizovaných řešení nebo při zavádění některé z cloudových služeb.
- Smlouva o mlčelivosti
  Pro ochranu obchodního tajemství (know-how) a důvěrných informací se zavazují obě strany k mlčenlivosti, ať už požadavek na utajení chtějí obě nebo jedna strana.

[5] Dílem se rozumí zhotovení určité věci (nespadá-li pod kupní smlouvu) a dále údržba, oprava nebo úprava věci nebo činnost s jiným výsledkem. (Jansa et al., 2018, s. 191).
[6] Smlouva o dílo není příliš vhodná pro agilní způsob vývoje softwaru.

- Servisní smlouva
  Pro údržbu softwaru je základním dokumentem servisní smlouva (Smlouva o poskytování servisních služeb k aplikačnímu programovému vybavení), která ustanovuje podmínky vzájemné dohody na provozu softwaru. Z toho vyplývá, že předmětem servisní smlouvy je zpravidla závazek poskytovat služby v rámci údržby a rozvoje aplikačního programového vybavení s cílem zhotovený software rozvíjet, udržovat a odstraňovat případné defekty nebo jiné vady. Obsahuje pravidla řízení incidentů a problémů. S tím se pojí i průběžné dodávání aktualizované funkční, technické a uživatelské dokumentace.
- Běžnou součástí servisní smlouvy je Smlouva o úrovni poskytovaných služeb (anglicky Service Level Agreement - SLA). Smlouva SLA je formálním vyjádřením vztahu mezi provozovatelem softwaru (poskytovatelem outsourcingu) a zákazníkem. Obsahuje nejen definici obsahu, kvantity a kvality služby, ale i definici metrik pro měření rozdílu mezi očekáváním zákazníka a skutečně poskytovanou službou, výši a frekvenci plateb, sankce a další ujednání důležitá pro plnění služby. Bez dohody SLA by byl outsourcing provozu softwaru nebo model ASP těžko představitelný.

3.2 Outsourcing vývoje softwaru – specifika, přínosy a rizika

Tato práce se zabývá částečným outsourcingem IS/IT a to outsourcingem ICT procesu - vývoje softwaru (IS/ICT), outsourcingem provozu a údržby softwaru (IS/ICT), která běží na infrastruktuře zákazníka a outsourcingem ICT zdroje a jeho údržby – nájmem IT odborníků.

Software může organizace vyvinout svými zaměstnanci nebo si ho nechat vyvinout na míru třetími osobami (programátory, jiným výrobcem softwaru). Předpokladem užívání tohoto softwaru a databází v něm obsažených je poskytnutí práva jej užít – licence.

Mezi hlavní příčiny využití outsourcingu vývoje softwaru patří:

- Nedostatek interních IT zdrojů a kapacity vývojového týmu,
- Získání specifických technických dovedností, erudice, nových odborných názorů nebo služeb, kterými organizace nedisponuje.

Kromě toho lze konstatovat, že organizace si může outsourcingem zajistit nejen potřebnou úroveň služeb pro softwarový projekt, ale i služby vývoje softwaru na špičkové, světové úrovni.

Mezi další přínosy z hlediska objednavatele patří:

- Zvýšení pružnosti IT zdrojů (to je v souladu s agilními přístupy),
- Časová flexibilita na straně externích vývojářů živnostníků,
- Lepší využití kapacit IT týmu (uvolnění interních IT zdrojů),
- Vyšší efektivita a motivace zkušených externích vývojářů (díky zkušenostem z mnoha softwarových projektů),
- Snížení provozních nákladů (úspora lidských zdrojů),
- Sdílení rizik.

V následujících kapitolách jsou shrnuty kritické faktory externí dodávky a rizika outsourcingu vývoje softwaru.

3.2.1 Kritické faktory externí dodávky

Převedení části IT na externí dodavatele není snadné a mohou se vyskytnout problémy. Na to je potřeba se připravit. Proto je nutné ve smlouvě s externím dodavatelem ukotvit všechny podstatné náležitosti spolupráce. Klíčovým předpokladem pro využívání služeb třetích stran je to, že to, co činí organizaci jedinečnou, musí zůstat uvnitř v organizaci.

Při outsourcingu je důležité rozumět byznysu externího dodavatele. Na straně dodavatele je nutné stanovit především správný obsah služby ve smyslu toho, že vytěsňovat má smysl to, co může dodávat ve standardizované podobě více zákazníkům, a zároveň, co může snadno customizovat a integrovat dle požadavků. Pokud jde o poskytnutí odborníků různých IT profesí, může je přidělovat více zákazníkům dle potřeby. Obecně kritické je určení vhodného objemu a jeho škálování v dostatečné kvalitě z hlediska dostupnosti, doby odezvy a bezpečnosti, přijatelné ceny a vhodné metriky. Pro hladkou spolupráci je třeba vybrat stabilního a spolehlivého dodavatele s dostatkem referencí a uzavřít s ním smlouvu. Vyžaduje to i transfer znalostí o produktech a službách organizace na dodavatele a jeho personál. A v neposlední řadě je třeba specifikovat znalosti, které musí zůstat u zákazníka (Bruckner et al. 2012, s. 93).

Co se týká strany objednavatele v rámci částečného outsourcingu IS/ICT, je třeba analyzovat kritické faktory pro organizaci, a to především v oblastech aplikační a technologické architektury IS/ICT a rozhraní mezi komponentami. Dále je třeba zohlednit optimální granularitu architektury, znalosti vlastních zaměstnanců a integraci celého IS/ICT, která zůstává v zodpovědnosti organizace. To znamená, jaké z toho plynou nároky na interní specialisty, přičemž náročnost dramaticky roste s růstem počtu externích dodavatelů. Objednavatel musí provádět i detailní controlling nákladů, analyzovat detailní informace o trhu a podle toho vybírat komponenty k outsourcingu. SLA se musí soustředit na popis rozhraní mezi poskytovatelem a objednavatelem, nikoliv jen na způsob zabezpečení těch zodpovědností, které na poskytovatele přešly. Je třeba vybrat optimální sadu metrik pro měření zvoleného outsourcingového vztahu. Je třeba monitorovat případné napojení provozního personálu poskytovatele na IS zákazníka (přes vystavené API rozhraní).

Vzhledem k cílům práce se budu zabývat jen vývojem softwaru na zakázku. V případě přesunutí vývoje softwaru na externí třetí stranu musí objednavatel dát pozor na kvalitní definici požadavků na vyvíjený software v průběhu projektu, bezproblémovou integraci dodaného softwaru do IS/ICT, předání souvisejících znalostí a dovedností od dodavatele a v neposlední řadě na flexibilitu dodavatele při řešení změn.

Pro obsazení případných volných pozic je možné outsourcing nakupovat. Flexibilně jednoduše. Odborníci mohou být najímaní přes agentury jako Bodyshopper nebo jako externisti na servisní smlouvu.

Organizace mohou nakupovat služby vývoje softwaru buď od dodavatelů působících v zemi jejich původu, nebo od dodavatelů ze zahraničí (anglicky offshore outsourcing).

Tvorba softwaru může být v místě zákazníka, vzdáleně nebo kombinovaně. Stejně tak to platí pro podporu a údržbu aplikací. Naproti tomu služby řízení infrastruktury mohou být poskytovány jen v místě zákazníka nebo vzdáleně.

3.2.2 Rizika spojená s outsourcingem vývoje softwaru

Vytěsnit části IS/IT bez rizika dost dobře nejde. Cílem této části je zmapovat rizika spojená s outsourcingem vývoje, implementace a provozu softwaru.

Rizika jsou určitá forma nebezpečí. Jsou to ale pouhé možnosti, u kterých je možné odhadnout pravděpodobnost a dopad, a s těmi lze pracovat a připravit se na ně.

Úkolem CIO a CTO je rovněž zvážit negativní důsledky především ve ztrátě konkurenční výhody, kterou může organizace získat díky svým IT expertům.

Před následujícími riziky z částečného vytěsnění IS/IT je třeba se mít na pozoru:

- Riziko závislosti na dodavateli softwaru či poskytovateli,
    - pokud dodavatel přestane plnit závazky, může to mít fatální následky pro zákazníka
- Riziko úpadku dodavatele softwaru,
- Riziko nesplnění smluvních požadavků,
- Riziko nesplnění podstatných požadavků na software,
    - v důsledku neodlišení důležitých požadavků od těch méně významných požadavků
- Riziko špatného poskytování služby, tedy nekvalitního softwaru s chybami,
    - nedostatečná dostupnost, odezva (výkon) a vyrovnávání se se zatížením (škálovatelnost)
- Riziko nedostatečné odolnosti dodaného softwaru vůči útokům z internetu – typicky u webových aplikací,
    - zranitelnost softwaru vůči útokům - zapojit bezpečnost do vývoje, a to do celého životního cyklu aplikace, a kryptografické zabezpečení
- Riziko úniku a zneužití důvěrných dat (informací) objednavatele,
    - důvěrné informace mohou být zneužity pro získání konkurenční výhody
- Riziko ztráty know-how ve firmě objednavatele,
- Riziko zneužití zdrojových kódů – toho se obávají obě strany,
- Riziko vzniku škody například neposkytnutím součinnosti nebo nepředvídatelná škoda,
    - sjednat limitaci náhrady škody
- Riziko technického výpadku (v provozu).

Existují ještě další rizika spojená s projektovým řízením, ta ale nejsou zahrnuta v rozsahu této práce.

Lze konstatovat, že riziko roste s mírou závislosti na externích dodavatelích softwaru. Cesta, jak se vyhnout zbytečným rizikům, je připravit buď smluvní pojistky pro každé riziko nebo opatření k jejich zmírnění nebo rovnou eliminaci.

Mezi obecné účinné záruky, kterými je možné zmírnit výše uvedená rizika, patří dobré jméno dodavatele, jehož poškození by vedlo k ztrátě pozice na trhu, pojištění u třetího subjektu (přenos rizika na třetí subjekt) a investiční účast dodavatele například na základě smlouvy o tichém společenství. Mimo rozsah této práce je rovněž uvedení všech opatření k eliminaci či zmírnění rizik.

4 Návrh souboru zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru

Tato kapitola se věnuje hlavnímu cíli mé diplomové práce, kterým je návrh souboru zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru a návrh vhodných modelů týmové organizační struktury. Zároveň slouží k dosažení jednak dílčího cíle práce **DC4** v podkapitole 4.2 a jednak dílčího cíle práce **DC5** v podkapitole 4.3.

Věřím, že přístup DevOps může být úspěšný, i pokud organizace využívá externí outsourcing pro vývoj softwaru nebo provoz softwaru. Jen to vyžaduje velkou pozornost od vedoucích pracovníků IT oddělení. Jedna z cest, jak propojit oba koncepty, je stanovit soubor pravidel, kterými je dobré se řídit.

Vzhledem k hlavnímu cíli práce jsem vypracoval soubor zásad pro propojení konceptů DevOps a outsourcingu funkčních oblastí vývoje softwaru nebo jeho provozu, které jsou uvedené v této kapitole. Motivací k uplatnění zásad je často neuspokojivá spolupráce a nespokojenost obou partnerů. Dalším důvodem jsou přínosy, kterých mohou oba partneři dosáhnout, pokud se jimi budou řídit a dodržovat je. Přínosy zavedení zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru jsou shrnuty v kapitole 4.2.2.

V této kapitole se zabývám také vlivem organizačních změn na týmové organizační struktury v důsledku zavedení přístupu DevOps a tím i na změny řízení organizace.

Ze vztahu mezi organizací a dodavatelem vyplývají následující role:

- Objednavatel
  Objednatel je v podstatě investor, který nese ekonomické náklady realizace softwarového projektu. Zároveň tento investor outsourcuje vývoj softwaru, jeho provoz nebo obojí.
- Externí dodavatel
  Externí dodavatel je zhotovitel softwaru poskytující licenci, který stává externím partnerem zajišťující vývoj softwaru. Mohou jim být softwarové společnosti nebo i jednotlivci (živnostníci). Z hlediska vývoje softwaru je externí dodavatel třetí stranou[7].

Soubor zásad se vztahuje přímo na organizace, které vystupují v roli objednavatelů, a externí dodavatele, kteří jim poskytují své služby. Na dodávce a provozu softwaru spolupracují i další zúčastněné strany: poskytovatelé služeb, partneři, klienti, zákazníci a interní zaměstnanci. Na tyto zúčastněné strany se zásady vztahují nepřímo.

Pro navržené zásady byly definovány následující výchozí předpoklady, za kterých zásady platí:

- Na straně objednavatele se předpokládá, že má už zavedené některé z vlastností DevOps a preferuje agilní vývoj softwaru.
- Dále se předpokládá u objednavatele, že má odborníky na vývoj softwaru a jeho provoz.

[7] Třetí stranu definuji v této práci ve smyslu, že organizace je jedna strana, vlastní zaměstnanci nebo zákazníci jsou druhá strana a externí subjekty jako jsou externí dodavatelé, poskytovatelé služeb jsou třetí strana.

- Objednavatel je vybaven moderními vývojovými nástroji, se kterými může pracovat externí dodavatel nebo je schopen se flexibilně dovybavit těmito nástroji.

Kromě výchozích předpokladů mají zásady následující omezení:

- Z důvodu omezeného rozsahu práce byly zásady formulované pro střední až větší organizace nebo banku.

Všechny zásady je třeba ověřovat, vyhodnocovat a zdokonalovat v průběhu spolupráce s externím partnerem na softwarových produktech nebo službách. Za to jsou zodpovědní pro obě skupiny zásad (společné a interní) jednak vedoucí pracovníci objednavatele a jednak samotné multidisciplinární týmy, které mají plnou odpovědnost od začátku myšlenky či konceptu až po ukončení produktu nebo služby, jak vyplývá z DevOps principu Celostní odpovědnosti. Vedle objednavatele je za společnou skupinu zásad odpovědný také externí dodavatel.

U každé zásady je uveden způsob, jak má být zásada vyhodnocena. Způsob hodnocení zásad je popsán v kapitole *4.1 Hodnocení zásad*.

4.1 Hodnocení zásad

V této podkapitole je popsán způsob hodnocení navržených zásad vedoucími pracovníky IT oddělení objednavatele. To je základní předpoklad pro jejich ověření v kapitole 5. Uplatňování zásad má přispívat u obou partnerů k postupnému neustálému zlepšování v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace. Jedině díky vzájemné komunikaci vnímaných odchylek od dohodnutých zásad a zpětné vazbě je možné procesy a úroveň spolupráce kontinuálně zlepšovat

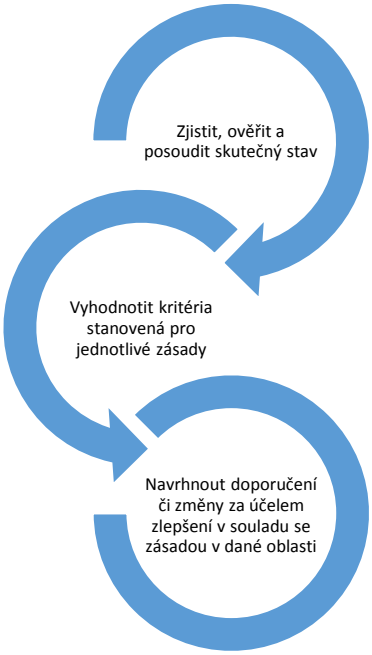
Za tímto účelem navrhuji použít následující klasické hodnocení. Nicméně použity mohou být i jiné metody, jako například PDCA cyklus (anglicky Plan-Do-Check-Act). To znamená v češtině „Plánuj, udělej, zkontroluj, uskutečni. Jsou to základní kroky pro dosažení neustálého zdokonalování.

Hlavními kroky hodnocení pro ověření zásad jsou:

1. Zjistit, ověřit a posoudit skutečný stav
2. Vyhodnotit kritéria stanovená pro jednotlivé zásady
3. Navrhnout doporučení či změny za účelem zlepšení v souladu se zásadami v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace

Princip hodnocení zásad je znázorněn na následujícím obrázku.

Obrázek 4.1 – Hodnocení zásad



Zdroj: vlastní zpracování

1. Zjistit, posoudit a ověřit skutečný stav
   Abychom byli schopni určit stav, ve kterém se organizace aktuálně s vývojem software a IT provozem nachází, musí proběhnout dialog, objektivní hodnocení a měření s týmy podílejícími se na vývoji nových aplikací - měření zralosti (anglicky maturity) v oblasti agilních metodik, kultury, organizace, dodávání produktů, procesů DevOps a automatizace. Hodnocení a měření by mělo dopomoci k určení slabých míst ve vývojovém procesu, na kterých je nutné zapracovat a vnést do nich více praktik z DevOps a Lean, než tomu bylo dosud.
2. Vyhodnotit kritéria stanovená pro jednotlivé zásady
   Cílem této fáze je vyhodnotit poznatky z předchozí fáze nebo z ověřovacího provozu v jednotlivých oblastech a porovnat je, zda jsou v souladu s navrženými zásadami, postupy plynoucími z přístupu DevOps a agilních rámců pro vývoj softwaru. Za tímto účelem jsou dva z agilních rámců popsané v kapitole 2.2. Vyhodnocení probíhá na základě aplikace předem stanovených kritérií na zjištěný stav. Jejich účelem je měřit, zda jsou zásady splněny.
3. Navrhnout doporučení či změny za účelem zlepšení v souladu se zásadami v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace
   Cílem této fáze je na základě výsledků předchozích fázích a v souladu se zásadami navrhnout změny, opatření a řešení pro zlepšení v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace, která bude možné zavést do praxe. V této fázi je nezbytná zpětná vazba a zapojení obou partnerů.

Popsaný postup hodnocení může probíhat iterativně, čímž je minimalizováno riziko spojené se zlepšováním ve všech oblastech formou „velkého třesku“.

Jako další fázi je možné si představit realizaci navržených opatření. Tato fáze je ale mimo rozsah této práce.

Obsah hodnocení jednotlivých zásad by se neměl měnit příliš často. Důvodem je zachování kontinuity hodnocení a sledování vývoje kritérií, jejich změny a trendy.

4.2 Soubor zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru

Organizace, která zavedla nebo se chystá zavést DevOps a zároveň si zajišťuje vývoj a provozování softwaru od externího partnera formou outsourcingu, se musí řídit zásadami pro přístup DevOps v podmínkách outsourcingu vývoje softwaru, které jsou popsány v této kapitole. Obsah této kapitoly přímo slouží k dosažení dílčího cíle práce **DC4**.

Zásady se dělí do dvou základních kategorií podle toho, zda se má jimi zabývat jak objednavatel, tak i externí dodavatel, tedy oba partneři, nebo jen objednavatel.

1. Skupina společných zásad jak pro objednavatele, tak i pro externího dodavatele
2. Skupina zásad pro objednavatele (interní)

Společné zásady řeší oboustrannou problematiku týkající se technických prekvizit, kompetencí, právního rámce a vytvoření prostředí pro produktivní spolupráci. Naproti tomu interní zásady řeší vnitřní úkoly, které si objednavatel musí splnit sám ve své organizaci. Interní zásady zahrnují zásady zralosti organizace a zásady řízení benefitů. Zásady zralosti organizace jsou zaměřeny na zlepšování vstupních kritérií, jako je výběr dodavatele, znalosti, míra automatizace, schopnost efektivního zapojení externího partnera a důvěryhodnost partnera. Naproti tomu zásady řízení benefitů jsou orientovány na zlepšování výstupních kritérií, jako jsou metriky pro měření výkonnosti IT procesů, řízení a správa dodávky softwaru, důvěryhodnost softwaru a controlling nákladů.

Pro přehlednost jsou zásady uvedeny v následujícím seznamu. Podrobně jsou pak popsány v podkapitole *4.2.1 Zásady pro úspěšné zvládnutí DevOps ve spolupráci s externím partnerem*. Popis všech zásad má jednotnou strukturu, která zahrnuje vysvětlení, význam a způsob hodnocení splnění zásady. Z hlediska významu se posuzuje, proč je důležitá a na co má dopady. Způsob hodnocení splnění zásady řeší otázku, jak bude posuzováno splnění zásady. U všech zásad je postaven na měření definovaných kritérií. Nicméně, tato práce se zabývá i hodnocením zásad v širším kontextu, který zahrnuje vedle vyhodnocení kritérií také posouzení skutečného stavu oblastí zájmu a návrh doporučení či změny za účelem zlepšení v souladu se zásadami v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace. Tento postup ověření zásad byl popsán v předchozí kapitole *4.1 Hodnocení zásad*.

Pro úspěšnou spolupráci objednavatele s externím dodavatelem v podmínkách outsourcingu vývoje softwaru byly formulovány následující zásady:

1. **Skupina společných zásad jak pro objednavatele, tak i pro externího dodavatele**
   Z01 - Zajistit si infrastrukturu jako službu - virtualizace a cloud computing
   Z02 - Využít maximálně principů a procesů DevOps na základě agilních přístupů
   Z03 - Usnadnit zahájení a ukončení spolupráce pro oba partnery

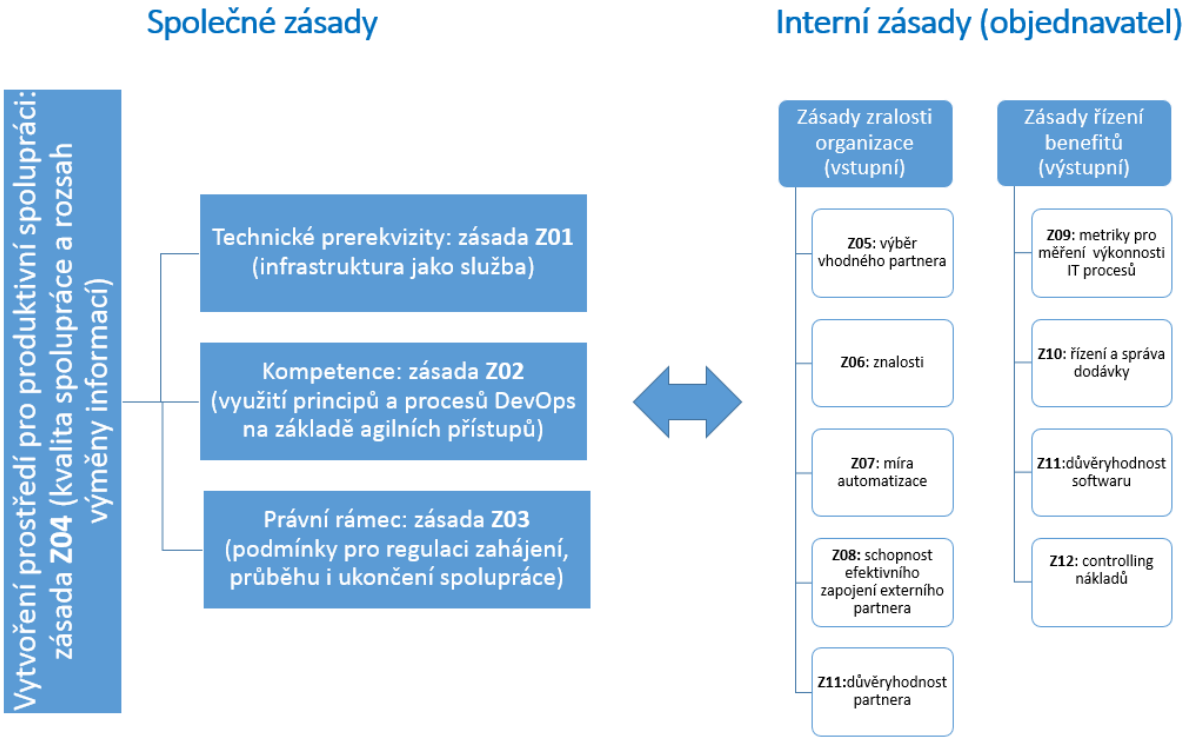
Z04 - Zkvalitnit spolupráci a rozsah výměny informací mezi partnery

2. **Skupina zásad pro objednavatele (interní)**
   Z05 - Vybrat vhodného partnera pro outsourcing
   Z06 - Zajistit znalost procesu softwarového vývoje v organizaci
   Z07 - Zajistit automatizaci rutinních procesů a zrychlit tok práce
   Z08 - Zajistit externím pracovníkům vývojových týmů přístup k interním prostředím a nástrojům
   Z09 - Definovat metriky výkonnosti IT v organizaci a vyhodnocovat rychlost, kvalitu a efektivitu dodání softwaru od externího partnera
   Z10 - Řídit a spravovat dodávku softwaru od externího dodavatele pomocí přístupů CI, CDE a CD
   Z11 - Zavést, spravovat a udržovat důvěryhodnost a bezpečnost softwaru na straně objednavatele
   Z12 - Zavést controlling nákladů

4.2.1 Zásady pro úspěšné zvládnutí DevOps ve spolupráci s externím partnerem

V této kapitole jsou charakterizovány jednotlivé zásady, jejich provázání a způsob jejich hodnocení. Postup hodnocení zásad se provádí ve třech krocích. Nejprve se zjistí, ověří a posoudí skutečný stav oblasti, poté se vyhodnotí kritéria stanovená pro jednotlivé zásady a na závěr jsou navržena doporučení či změny za účelem zlepšení dané oblasti. Postup hodnocení zásad je podrobně vysvětlen v kapitole *4.1 Hodnocení zásad*. Na následujícím obrázku jsou znázorněny vztahy mezi zásadami pro přístup DevOps v podmínkách outsourcingu vývoje softwaru.

Obrázek 4.2 – Vztahy mezi zásadami pro úspěšnou spolupráci v rámci outsourcingu vývoje softwaru



Zdroj: vlastní zpracování

Z01 - Zajistit si infrastrukturu jako službu - virtualizace a cloud computing

Tato zásada je založena na tom, aby si jak objednatel, tak i dodavatel pořídili infrastrukturu jako službu.

Virtualizace a cloud computing patří mezi klíčové faktory, které umožňují rozvoj DevOps. Techniky virtualizace umožňují jednoduše a efektivně spravovat infrastrukturu a oddělit od sebe jednotlivé prostředky (servery, aplikace, služby, sítě), aby se nemohly ovlivňovat. Základním stavebním prvkem infrastruktury je virtualizace serverů, na kterých jsou provozovány operační systémy a aplikace. Virtualizace infrastruktury usnadnila sdílení společných platforem a poskytování vývojových prostředí. Kromě toho vedla k rozvoji přístupů jako je kontejnerizace a cloud computing. Virtualizace služeb umožňuje více týmům vyvíjet a testovat software paralelně i bez sdíleného přístupu k interním systémům. Virtuální úložiště umožňuje týmům jednoduše si vyměňovat nebo sdílet data a dokumenty. Virtualizace sítí umožňuje vzdáleným týmům kódovat interní sítě. Využití virtuálních API a virtuálních privátních sítí je popsáno v zásadě *Z08 - Zajistit externím pracovníkům vývojových týmů přístup k interním prostředím a nástrojům*.

Cloud computing je model dodávání konfigurovatelných výpočetních prostředků, které zahrnují různé servery, aplikace, data a další prostředky, v integrované formě a poskytovaných jako služba přes internet (Bruckner et al. 2012, s. 68). V Cloud computingu je virtualizace prostředků běžná a je zajišťována externím poskytovatelem služeb.

Existují tři hlavní modely (Bruckner et al. 2012, s. 68): IaaS (Infrastructure-as-a-Service) pro přístup k výpočetnímu výkonu (serverům), úložištím a sítím přes internet, PaaS (Platform-as-a-Service) pro poskytování úplného výpočetního prostředí, které tvoří nejen stejné komponenty jako IaaS, ale i operační systém, middleware, runtime prostředí pro programy a další nástroje, a SaaS (Software-as-a-Service) pro poskytování aplikací přes internet. Další modely se vyvíjejí.

Model IaaS poskytuje větší volnost, ale mnoho toho nenabízí. V modelu SaaS jsou aplikace standardizované a jakékoliv zásahy v nich jsou velmi omezené. V podstatě jedině model PaaS umožňuje poskytovat vývojářům agilní infrastrukturu, kterou potřebují ke své práci. To znamená připravené prostředí a vývojové nástroje pro tvorbu aplikací a jejich hostování včetně dat. PaaS lze nastavit tak, aby podporoval celý životní cyklus aplikace. Z tohoto důvodu mohou týmy platforem v DevOps organizaci přijmout cloudové principy pro vývoj svých produktů, aby mohly Business týmům poskytnout virtualizovaná prostředí pro vývoj, testování a provoz „na kliknutí“ pomocí samoobslužných služeb, v rámci kterých si bude moci volit nastavení základních parametrů a povolených technologií.

V případě externího vývoje software je dobré se opřít o model PaaS a rozmyslet si, zda využít rozvinuté infrastruktury na straně poskytovatele či ji vybudovat na místě a poskytnout ji externímu partnerovi. Pro agilní vývoj je zásadní sjednotit infrastrukturu s dodavatelem.

Význam: Cloud computing a virtualizace poskytují agilní infrastrukturu, flexibilitu a škálovatelnost, které DevOps potřebuje k naplnění obchodních potřeb byznysu. Jsou předpokladem pro automatizaci poskytování infrastruktury Business týmům. V případě Cloud computing není potřeba zajišťovat vlastní prostory, nakupovat hardware nebo najímat další odborníky. Organizace se stará jen o aplikace a data.

Cloud computing určitě umožňuje zapojit dodavatele třetích stran do DevOps přístupu a využít jejich cloudových a hostitelských služeb, které dodají služby infrastrukturu na vyžádání.

Hodnocení zásady *Z01 - Zajistit si infrastrukturu jako službu - virtualizace a cloud computing* na základě následujících kritérií:

a) Z01K1 - Zavádění a využívání virtualizačních a cloudových technologií.
   Využívání virtualizačních a cloudových technologií je nutné škálovat podle typu aplikace.
   - Webové aplikace mohou plnohodnotně využívat cloudových služeb.
   - Naproti tomu u ERP aplikací je třeba pečlivě zvážit, zda je odsunout do cloudu.

   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z01K1O1 - Využívá organizace virtualizaci infrastruktury (serverů, úložišť dat)? Ano/Ne.
   - Z01K1O2 - Využívá organizace virtualizaci sítí? Ano/Ne.
   - Z01K1O3 - Využívá organizace virtualizaci služeb? Ano/Ne.
   - Z01K1O4 - Využívá organizace virtualizaci desktopů? Ano/Ne.
   - Z01K1O5 - Má organizace zavedené služby typu IaaS? Ano/Ne.
   - Z01K1O6 - Má organizace zavedené služby typu PaaS pro vývojové týmy? Ano/Ne.
   - Z01K1O7 - Využívá organizace služby typu SaaS? Ano/Ne.

b) Z01K2 - Poskytování prostředků infrastruktury pomocí samoobslužných služeb.
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z01K2O1 - Jsou vývojovým týmům v organizaci poskytovány prostředky infrastruktury pomocí samoobslužných služeb? Ano/Ne.

Z02 - Využít maximálně principů a procesů DevOps na základě agilních přístupů

Tato zásada říká, že objednavatel zavedl agilní přístupy a DevOps a to samé vyžaduje po externích dodavatelích. Je to o způsobu myšlení. Není možné akceptovat vodopádový přístup. Pro projekty vývoje softwaru musí být tedy použity agilní metodiky, například Scrum. Zároveň musí existovat zájem o přijetí DevOps konceptu (principů, kultury, procesů).

Pro splnění předpokladů použití agilních přístupů je třeba navázat na Manifest agilního vývoje a jeho položky. Z agilních metodik lze použít metodiku Scrum, protože je to prokazatelně úspěšná cesta, jak se stát agilními i ve spolupráci s externími dodavateli, neboť agilní IT firmy ji musí skvěle znát. Vývoj je iterativní, v každém cyklu se odbaví kousek softwaru (tzv. sprinty). Metodice Scrum je věnována kapitola *2.2.1 Scrum*. Pokud ji IT firma nezná nebo jsou agilní přístupy cizí, neměla by být taková firma hned vyřazena z výběrového řízení.

Procesy analýzy, vývoj, testování, nasazení, provozu a údržba musejí být prováděny podle DevOps principů a je třeba je neustále zlepšovat. Principům DevOps je věnována kapitola *2.4 Principy DevOps*.

Význam: Vývojový tým a odborníci z provozu se musejí řídit principem odpovědnosti od začátku do konce. Spolupráce na produktu či službě s nejlepším výsledkem pro zákazníka.

Hodnocení zásady *Z02 - Využít maximálně principů a procesů DevOps na základě agilních přístupů* na základě následujících kritérií:

a) Z02K1 - Soulad s procesy DevOps a agilními přístupy.
   Principy spolupráce s dodavatelem nastavuje objednavatel a rovněž je hodnotí.
   Měření pro toto kritérium je založené na porovnání jednotlivých procesů vývoje (včetně analýzy), integrace, testování, nasazení, provozu a údržby v organizaci s procesy DevOps, agilními přístupy a principy deklarovanými v Agilním Manifestu.
   - Z02K1O1 - Má dodavatel prokazatelnou zkušenost s principy a procesy DevOps?[8] Ano/Ne.
   - Z02K1O2 - Je proces vývoje (včetně analýzy) v souladu s DevOps a agilními přístupy a principy? Ano/Ne.
   - Z02K1O3 - Je proces integrace v souladu s DevOps a agilními přístupy a principy? Ano/Ne.
   - Z02K1O4 - Je testování v souladu s DevOps a agilními přístupy a principy? Ano/Ne.
   - Z02K1O5 - Je proces nasazení v souladu s DevOps? Ano/Ne.
   - Z02K1O6 - Jsou procesy provozu a údržby v souladu s DevOps? Ano/Ne.

b) Z02K2 - Spokojenost s výsledným softwarovým produktem nebo s přírůstky dodanými na konci sprintů ze strany objednavatele a ze strany zákazníků.
   Měření pro toto kritérium je založené:
   - Na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů (posloupnost od nejmenší k největší). Hodnocení bude spočívat v udělování bodů, přičemž hodnota 10 znamená „naprostá spokojenost“ (bez výhrad) a hodnota 0 znamená „absolutní nespokojenost“.

c) Z02K3 – Kvalita softwaru
   Pro měření je možné použít následující metriku chybovosti:
   - Počet reportovaných chyb, která je indikátorem kvality softwaru a rovnováhy mezi tvorbou softwaru a odchytáváním chyb.

Navrhnout doporučení či změny za účelem zlepšení v souladu s DevOps procesy a agilními přístupy.

Z03 - Usnadnit zahájení a ukončení spolupráce pro oba partnery

Odstoupení od smlouvy a ukončení spolupráce musí být snadno proveditelné na obou stranách za stanovených podmínek. Stejně tak musí být možné rychle zahájit spolupráci.

Z hlediska ukončení spolupráce má tato zásada preventivní charakter. Pokud nebude tvůrce softwaru odevzdávat tak, jak se dohodne s objednavatelem na konci sprintů, je to důvod k odstoupení od smlouvy a ukončení spolupráce. Na objednavateli pak je, aby si našel jinou IT firmu, která buď danou práci dokončí (měl by mít převzaté zdrojové kódy) nebo začne od začátku. Každopádně je třeba ctít agilní manifest, který říká, že je třeba "*Spolupráce se zákazníkem před vyjednáváním o smlouvě*". Na druhou stranu pokud objednavatel nezajistí nutnou součinnost, musí být

[8] Otázka Z02K1O1 souvisí se zásadou *Z05 - Vybrat vhodného partnera pro outsourcing.*

upozorněn a pokud nesjedná nápravu, musí být možné ukončit spolupráci i na straně tvůrce softwaru. Dále v případě pokud bude objednavatel spokojený s předaným softwarem na konci jakéhokoliv sprintu a řekne "stop, tohle už mi stačí", musí to dodavatel akceptovat a nepokračovat v práci, protože by mu nezaplatil.

Důležitá je i schopnost objednavatele identifikovat okamžik dodání softwaru, kdy už bude rozvíjet aplikaci vlastními silami bez externího dodavatele.

Význam: V případě nespokojenosti s kvalitou dodavatele, je nutné velmi rychle ukončit celý vývojový proces. Z praktického hlediska je možné uzavřít rámcovou smlouvu a dílčí smlouvy na jednotlivé sprinty (iterace) nebo je nutné sestavit smlouvu tak, aby dovolila odstoupení za jasných podmínek po každé iteraci. Tato možnost nutí externí tvůrce softwaru pracovat kvalitně a dodávat kvalitní aplikace, protože by jinak nemohl obstát na trhu. Čisté ukončení smluvního vztahu, bez ohledu na to, jak dlouho trval, by se mělo považovat za samozřejmost. Usnadní to situaci všem stranám. Na rozdíl od toho, když dojde ke sporům, které mohou vést až k soudu, které mohou trvat řadu let.

Hodnocení zásady *Z03 - Usnadnit zahájení a ukončení spolupráce pro oba partnery* na základě následujících kritérií:

Za prvé charakter smluv musí reflektovat agilní vývoj softwaru a umožnit snadné odstoupení od smlouvy a ukončení spolupráce v případě nekvalitního dodavatele například na konci sprintu. Za druhé je potřeba se vyvarovat se obecných ustanovení, které nic neřeší, ale přesně formulovat ustanovení týkající se odstoupení od smlouvy, aby byla vymahatelná a eliminovalo se riziko soudního sporu. Doporučit lze konzultaci smlouvy se zkušenými IT odborníky a právníky, kteří se zabývají IT právem.

a) Z03K1 - Odstoupení od smlouvy a ukončení spolupráce
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z03K1O1 - V případě nekvalitního dodavatele je možné snadno odstoupit od smlouvy na vývoj softwaru na konci iterace (sprintu)? Ano/Ne.
   - Z03K1O2 - Jsou smluvní ustanovení ohledně odstoupení od smlouvy precizně formulovaná? Ano/Ne.
b) Z03K2 - Schopnost identifikovat okamžik dodání softwaru
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z03K2O1 - Sleduje organizace naplnění podstatných požadavků s hodnotou pro zákazníka (například vůči business case v RFI)?
   - Z03K2O2 – V případě naplnění podstatných požadavků s hodnotou pro zákazníka (splnění business case) je schopen objednavatel převzít dodávku softwaru a rozvíjet ho bez externího dodavatele?

Z04 – Zkvalitnit spolupráci a rozsah výměny informací mezi partnery

Turbulentní prostředí způsobuje časté změny a malé týmy pracující na softwarovém projektu se ukazují jako velmi efektivní. DevOps a agilní přístupy podporují větší intenzitu interakce a efektivitu spolupráce mezi byznysem, vývojáři a odborníky z provozu. Vývojové týmy předvádějí v rámci pravidelných iterací výsledky své práce zákazníkovi a vizualizují je formou funkčních řešení

a prototypů. Zákazník je v centru dění a dává vývojovému týmu zpětnou vazbu. Klíčové je sledování business hodnoty a rizika. Je třeba očekávat, že může kdykoliv své požadavky změnit například kvůli konkurenci. To vyžaduje nejen vytvoření informačních a komunikačních kanálů, kterými se dostanou informace včas ke správným lidem, i společných týmů s příjemci řešení. Zpětná vazba musí být integrována do jejich každodenní práce.

Velkým přínosem agilní metodiky Scrum je transparentnost a to, zapojuje všechny hlavní účastníky. Postup tvorby softwaru, a kam směřuje, je tak všem jasný a vládne o něm hluboké porozumění. To je klíčové pro kontrolu a přizpůsobování. Agilní praktiky jako je Daily Scrums zlepšují komunikaci, eliminují další schůzky, identifikují překážky dalšího rozvoje, zdůrazňují a podporují rychlé rozhodování a zlepšují úroveň znalostí vývojového týmu. Další oblíbenou vizualizační pomůckou, který slouží týmu ke sdílení informací a který pomáhá informovat, kdo na čem pracuje, co je již hotové a co zbývá dokončit, je Scrum tabule. Zpravidla je rozdělena do tří sloupců: backlog, nesplněné úkoly a hotovo. Tým zaznamenává každý den aktuální stav úloh do příslušných sloupců Scrum tabule. Prostřednictvím výměny informací se k členům týmu dostávají nejlepší postupy v oboru softwarového vývoje a zkušenosti externích dodavatelů získaných během práce na mnoha softwarových projektech.

Dalším užitečným procesním nástrojem, který umožňuje vizualizovat pracovní postup na základě rozpadávání pracovních úkolů na menší části a zobrazení na tabuli, je Kanban. Princip Kanban tabule je popsán v kapitole *2.2.2 Kanban*.

V případě distribuovaných týmu v různých zemí je potřeba počítat s tím, že komunikace s dodavatelem na dálku je vždy náročná a může vést ke zvýšení nákladů. Jednotlivé týmy spolu musí pracovat jako jednotný celek, sdílet znalosti, poskytovat vzájemnou zpětnou vazbu a nezavádět kulturu vzájemného obviňování se při neúspěchu. V takovém případě je velmi důležité, aby si každý tým dobře stanovil pravidla fungování a komunikace, aby lidé dobře plánovali svůj čas různých časových zónách a byli transparentní vůči ostatním. Tým může být distribuovaný i lokálně například z důvodu práce z domova. V obou případech je třeba pořádat videokonference místo klasických pracovních porad. Šetří čas a cestovní náklady. Podle studie společnosti Verizon se náklady vyjádřené v počtu pracovních hodin jednotlivých zaměstnanců snižují až o dvě třetiny ve chvíli, kdy schůzka probíhá přes videokonferenční techniku.

Pro úspěšnou spolupráci je důležité alespoň jedno osobní setkání s externisty nebo specialisty externí organizace. Tým vývoje a tým provozu mohou uspořádat neformální konferenci.

Nicméně důležité je oboustranné zachování utajení obchodních tajemství a důvěrných informací.

Význam: DevOps se snaží o zboření bariér mezi vývojem a IT provozem a klade důraz na otevřenou komunikaci a spolupráci. V podmínkách změn a nejistoty jsou úspěšnější menší agilní týmy, ve kterých jsou i integrováni i příjemci řešení, tj. zákazníci a zástupci objednavatele. Při komunikaci je třeba dbát na to, aby tým přesně věděl, co se od něj očekává, a aby měl neustále aktuální informace. Zároveň musejí využít existující nástroje pro komunikaci a spolupráci, aby experimentovali a vytvářeli nové nástroje, kterými si mohou vzájemně pomáhat. Důležité je zajistit předání znalostí interním zaměstnancům ať už smluvně, tak i podporou v rámci týmové spolupráce.

Hodnocení zásady *Z04 - Zkvalitnit spolupráci a rozsah výměny informací mezi partnery* na základě následujících kritérií:

a) Z04K1 - Spokojenost pracovníků
   Pro měření tohoto kritéria je možné použít měkké (anglicky soft) metriky jako je míra spokojenosti pracovníků, pro níž lze definovat následující způsob měření:
   - Spokojenost pracovníků měřit na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů (posloupnost od nejmenší k největší). Hodnocení bude spočívat v udělování bodů, přičemž hodnota 10 znamená „naprostá spokojenost" (bez výhrad) a hodnota 0 znamená „absolutní nespokojenost".

b) Z04K2 - Využívání restrospektivy a zpětné vazby pro zlepšování v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace.
   Měření pro toto kritérium je založené na odpovědi na otázku:
   - Z04K2O1 - Používá organizace v softwarových projektech restrospektivu (jako u agilního rámce Scrum) a zpětnou vazbu za účelem podpory komunikace a dosažení zlepšování v oblastech: spolupráce, produktu a procesu? Ano/Ne.

b) Z04K3 – Schopnost reprioritizace požadavků v rámci sprintu
   Jedná se především o schopnost pružné reprioritizace požadavků nebo dodávek v rámci sprintu s ohledem na potřeby objednatele.
   Měření pro toto kritérium je založené na odpovědi na otázku:
   - Z04K3O1 – Je objednavatel schopný pružné reprioritizace požadavků nebo dodávek v rámci sprintu s ohledem na své potřeby? Ano/Ne.

c) Z04K4 - Výměna poznatků mezi partnery
   Měření pro toto kritérium je založené na odpovědi na otázku:
   - Z04K4O1 - Probíhá mezi partnery výměna poznatků? Ano/Ne.

Z05 - Vybrat vhodného partnera pro outsourcing

Softwarové společnosti, které nabízejí služby softwarového vývoje, mají zpravidla lepší znalosti softwarového vývoje, řešení architektury a technologických možností (anglicky technology stack). Kromě toho dbají na svou reputaci, mají profesionální přístup a vyšší produktivitu, snadněji se u nich vynucuje kvalita softwaru a termíny. Dobře vybraný kvalitní dodavatel může urychlit jak dobu uvedení na trh (anglicky time to market), tak i reakce na změny, a snížit rizika.

Jak ale najít tu pravou softwarovou společnost, jejíž služby využijeme? Výchozím předpokladem je ujasnit si představu, čeho je třeba dosáhnout, a kvalitně připravit výběrové řízení. Vzhledem k DevOps principu celostní odpovědnosti je třeba nad výběrem dodavatele uvažovat z dlouhodobého hlediska a zapojit ho i do údržby a podpory.

Postup pro určení dodavatele může probíhat analogicky fázím, kterou jsou popsané v kapitole *3.1.2. Výběrové řízení*. Jen s tím rozšířením, že se nejprve vytipují IT firmy, které pracují agilně a které jsou případně schopny pracovat v režimu DevOps. Dodavateli se na začátku definuje jen záměr, vize, nanejvýš „big picture" či „hi-level" představa softwaru. Nedělá se předběžná analýza. Dodavatel musí splňovat podmínku agilního vývoje, to znamená, že práce na softwaru musí probíhat iterativně, kdy se v každém cyklu iterace odbaví určitá část softwaru, přičemž se nejdříve

realizují nejpodstatnější požadavky, které přinášejí největší hodnotu zákazníkovi, a na konci každé iterace musí být předán software v produkční kvalitě. Hodnotící kritéria mohou být stanovena na základě referencí, prokazatelných zkušeností s agilními přístupy a DevOps, ceně za jednotku práce (MD), odborné znalosti ve funkční oblasti softwaru, schopnosti dodat veškeré potřebné profese a Scrum mastra (pokud objednatel nemá vlastního) s certifikacemi a záruce pro zajištění bezpečnosti. Použití softwaru, hardwaru a služeb nakoupených od třetí stran nesmí ohrožovat bezpečnost. Účinnými argumenty jsou především zaměření dodavatele na agilní vývoj (čili jeho firemní kultura), jeho dobré jméno, důvěryhodnost, referenční zákazníci, u kterých je možné si ověřit zkušenosti s dodavatelem, a jeho velikost. Příliš malý dodavatel nemusí mít dostatek lidí po celou dobu projektu. A pro příliš velkého dodavatele to nemusí být finančně zajímavé natolik, aby do projektu přidělil své nejlepší lidi.

S úspěšným dodavatelem ve výběrovém řízení je potřeba uzavřít spolupráci na základě rámcové smlouvy s možností uzavírat dílčí smlouvy na poskytnuté služby, neboť tato kombinace je nejvhodnější pro agilní vývoj, jak je uvedeno v kapitole *3.1.3. Smluvní vztahy*. Zároveň musí umožňovat snadné odstoupení od smlouvy, jak požaduje zásada *Z03 - Usnadnit zahájení a ukončení spolupráce pro oba partnery*.

Význam: Celý proces určení dodavatele pro outsourcing vývoje softwaru usnadnit a zvládnout rychle. To umožní urychlit nastartování softwarového projektu. V souladu s agilními přístupy musí mít dodavatel zájem na to, aby vyprodukoval něco, z čeho objednavatel či zákazník bude mít užitek. Klíčová je dobrá spolupráce se zákazníkem ze strany dodavatele, zpětná vazba, zlepšování práce a předání znalostí. Proto je potřeba s ohledem na DevOps principy celostní odpovědnosti a nestálého zlepšování, aby objednatel usiloval o dlouhodobý vztah s externím dodavatelem softwaru. Ze strany objednavatele je podstatné využít nejlepších postupů agilního vývoje softwaru od externího partnera.

V případě dobrých zkušeností s dodavatelem z hlediska kvality a dodržování termínů je možné stejného partnera využít pro další softwarové projekty. Pokud objednatel bude s dodavatelem nespokojený, příště si ho neobjedná.

Hodnocení zásady *Z05 - Vybrat vhodného partnera pro outsourcing* na základě následujících kritérií:

a) Z05K1 - Spolupráce dodavatele
   Měření pro toto kritérium je založené:
   - Na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů měřit spolupráci dodavatele (posloupnost od nejmenší k největší). Hodnocení bude spočívat v udělování bodů, přičemž hodnota 10 znamená „naprostá spokojenost“ (bez výhrad) a hodnota 0 znamená „absolutní nespokojenost“.

b) Z05K2 - Spokojenost s dodavatelem
   Měření pro toto kritérium je založené:
   - Na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů měřit spokojenost s dodavatelem (posloupnost od nejmenší k největší). Hodnocení bude spočívat v udělování bodů, přičemž hodnota 10 znamená „naprostá spokojenost“ (bez výhrad) a hodnota 0 znamená „absolutní nespokojenost“.

Z06 - Zajistit znalost procesu softwarového vývoje v organizaci

Tato zásada říká, že objednavatel musí skvěle znát své procesy softwarového vývoje, testování, nasazování a provozu a nástroje jejich podpory, aby je mohl prezentovat externímu dodavateli. Znalost procesů a nástrojů objednavatele je základním předpokladem pro externího dodavatele, pokud má objednavateli dodávat software.

Na druhou stranu v agilním světě není povinné mít striktně definované a řízené procesy. Místo toho agilní metody jen vymezují hřiště. Nutné je respektovat agilní manifest, který říká *„Jednotlivci a interakce před procesy a nástroji"*.

Z hlediska nastavení procesů je třeba varovat před situací, která může v organizacích nastat, že všichni fungují přesně podle procesů, ale nic není hotové. Pro tento případ je třeba uplatňovat zásady *Z02 - Využít maximálně procesů a principů DevOps na základě agilních přístupů* a *Z09 - Definovat metriky výkonnosti IT v organizaci a vyhodnocovat rychlost, kvalitu a efektivitu dodání softwaru od externího partnera* a princip Neustálého zlepšování.

Dále je třeba mít nastavené adekvátní procesy pro řešení situací, které mohou ve vztahu k externím dodavatelům nastat.

Význam: Úspěch DevOps závisí na dobrých procesech. Řízení podpůrných IT procesů odhaluje rutinní procesy a ty je možné automatizovat. Pro neustálé zlepšování je klíčové mít zmapované procesy, znát jejich průběh a chápat příčiny a dopady procesních změn, aby bylo možné urychlit každý jejich krok a odstranit čekání, tj. jeden z typů odpadů při vývoji softwaru. Procesy lze podporovat různými nástroji. Následná automatizace procesů podporuje agilní vývoj a zaměření na sledování hodnoty pro zákazníka.

Hodnocení zásady *Z06 - Zajistit znalost procesu softwarového vývoje v organizaci* na základě následujících kritérií:

Na úvod je nutné konstatovat, že komplexní hodnocení zralosti procesů přesahuje rozsah této práce. Stejně tak diskuze, zda je žádoucí hodnocení zralosti procesů agilního způsobu vývoje softwaru.

a) Z06K1 - Znalost IT procesů
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z06K1O1 - Má organizace popsány IT procesy? Ano/Ne.
   - Z06K1O2 - Je organizace schopná prezentovat procesy vývoje softwaru a jeho dodávky externímu dodavateli? Ano/Ne.

b) Z06K2 - Proces vývoje softwaru
   Měření pro toto kritérium je založené na:
   - Porovnání procesu vývoje softwaru v konkrétním softwarovém projektu nebo v organizaci s agilním způsobem vývoje softwaru například na základě agilních metodik Scrum a Kanban.

Z07 – Zajistit automatizaci rutinních procesů a zrychlit tok práce

Automatizace je jedním z pilířů DevOps prostředí, jak je uvedeno v kapitole *2.5 Holistický přístup DevOps od vývoje po provoz softwaru*. DevOps umožňuje zavést správné procesy ve vývojovém cyklu softwaru a automatizace dokáže dobře navržené, standardizované procesy zrychlit a odstranit překážky. V rámci DevOps se automatizují procesy integrace, sestavování, testování a nasazování softwaru. Kromě toho i se používá i automatizované poskytování. Podstatné přitom je zajistit dodržování stanoveného systému nasazování a kontroly verzí programového kódu.

Pro podporu automatizace musí organizace spolu s externím partnerem zavést, nastavit a důsledně používat automatizační nástroje, jejichž možnosti se neustále rozšiřují s vývojem technologií. Jedině tak bude možné viditelně sledovat průběh procesů v každodenních vývojových aktivitách.

Na druhou stranu automatizované procesy musejí zůstat adaptativně přizpůsobitelné dle potřeb vývojových a provozních týmů. V případě použití více automatizačních nástrojů v organizaci je třeba zajistit jejich orchestraci.

Význam: Smyslem automatizačních přístupů je zbavit se rutinních a opakujících se činností proto, aby lidé měli více času věnovat se na jedné straně rychlé opravě defektů a na druhé koncepční a tvůrčí činnosti, tj. návrhu a tvorbě softwaru, rozšiřování funkcionality a rozvoji aplikací podle požadavků zákazníka. To platí i pro třetí strany. Vedle snížení nákladů přispívají k zajištění kvality, zrychlení a nasazování velkého množství měnícího se kódu. Vyšší kvality dosahují vyloučením lidských chyb, kterého mohou nastat v případě ručního provádění kroků, a dřívějším odhalováním vad softwaru na základě častější frekvence spuštění automatizovaných testů. Důležité obsáhlé regresní testy a jejich vyhodnocení mohou běžet přes noc a jejich výsledky mohou být k dispozici druhý den. Výsledkem je méně zavlečených chyb do produkce, které se například nestihnou odhalit a opravit v releasu.

Hodnocení zásady *Z07 – Zajistit automatizaci rutinních procesů a zrychlit tok práce* na základě následujících kritérií:

a) Z07K1 - Automatizace procesů
   Měření pro toto kritérium je založené podle typu aplikace na odpovědích na otázky:
   - Z07K1O1 - Jsou všechny podstatné procesy natrvalo automatizované? Ano/Ne.
   - Z07K1O2 - Existuje podpora pro automatizaci procesů? Ano/Ne.

   Za druhé na posouzení míry automatizace procesů integrace, testování a nasazování softwaru v rámci softwarového projektu a organizace ve srovnání s DevOps a vysoce výkonnými IT společnostmi.
b) Z07K2 – Průchodnost kódu do produkce
   Pro měření je možné použít metriky průchodnosti kódu a porovnat je s vysoce výkonnými IT společnostmi (například na základě ročních reportů organizace Puppet):
   - Frekvence nasazování,
   - Doba potřebná k realizaci změny,
   - Doba potřebná do opravy chyby v produkci s ohledem na její dopad.
c) Z07K3 – Kvalita softwaru (chybovost)
   Pro měření je možné použít následující metriku chybovosti změn a porovnat ji s vysoce výkonnými IT společnostmi (například na základě ročních reportů organizace Puppet):
   - Míra selhání změn.

Z08 – Zajistit externím pracovníkům vývojových týmů přístup k interním prostředím a nástrojům

Tato zásada je na obrázku *Zásady úspěšné spolupráce* označena jako „schopnost efektivního zapojení externího partnera".

Virtualizační technologie umožňují zapojit externí pracovníky vývojových týmů do interního prostředí a sdílet je s nimi, ať už jsou jen přes ulici nebo pracují po celém světě. V tomto kontextu je vývoj softwaru proces, který lze provádět vzdáleně a výsledky poskytovat prostřednictvím síťové infrastruktury. Zároveň zapojení externích pracovníků vývojových týmů včetně zřízení přístupů a případných předepsaných školeních, aby se externí vývojář mohl co nejdříve pustit do práce.

V případě vývoje softwaru na zakázku je sdílený přístup k interním prostředím a nástrojům zásadní. Interními prostředími a nástroji jsou myšleny především vývojová prostředí a vývojové nástroje. V případě agilního vývoje musí objednatel počítat s tím, že musí IT firmě zpřístupnit svou infrastrukturu a interní systémy včetně testovacích dat a přístupových údajů. Na druhou stranu musí chránit přístup k důvěrným informacím. Základem je důvěryhodnost externího partnera a to úzce souvisí se zásadou *Z11 - Zavést, spravovat a udržovat důvěryhodnost a bezpečnost softwaru na straně objednavatele.*

IT firmy využívají následující formy vzdáleného přístupu ke zdrojům (aplikacím):

- Virtuální API – tato externí rozhraní umožňují vzdáleným odborníkům přistupovat k vystaveným funkcím nebo microservisám a pracovat nezávisle, aniž by přitom ovlivňovali práci ostatních.
- Virtuální privátní sítě (VPN) - umožňují bezpečný přístup k interním systémům pro vzdálené uživatele odkudkoliv v dosahu internetu. Tunelové připojení VPN chráněné dostatečně silným šifrováním je dominantní forma vzdáleného přístupu.
- VDE – virtuální stanice - umožňují přístup do virtuálního prostředí. Pro VDE je nutné zajistit dostatečnou latenci.

Součástí přístupu k VPN síti a k VDE může být i zajištění požadované autentizace, neboli ověření toho, že uživatel je skutečně tím, za koho se vydává.

Význam: Schopnost rychlé efektivní zapojení externího partnera je nutným předpokladem pro agilním způsob vývoje softwaru. Externí dodavatel musí zajistit, že k systémům, na kterých má být provozována aplikace, mohou mít přístup jen ověření a oprávnění pracovníci, to znamená jen ti, kteří mají schválený přístup

Hodnocení zásady *Z08 - Zajistit externím pracovníkům vývojových týmů přístup k interním prostředím a nástrojům* na základě následujících kritérií:

a) Z08K1 - Sdílení interních prostředí a nástrojů s externím dodavatelem
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z08K1O1 - Sdílí organizace svá interní prostředí a nástroje s externími partnery? Ano/Ne.
   - Z08K1O2 - Je možné nastavit sdílení interních prostředí a nástrojů během jediného pracovního dne? Ano/Ne.

b) Z08K2 - Používané formy vzdáleného přístupu, jejich kvalitu a bezpečnost například formou interního IT auditu. Posoudit u jednotlivých forem, jak kvalitně může externista pracovat bez technických prodlev, výpadků techniky a sítí.
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z08K2O1 – Používají externí partneři organizace VPN pro přístup do interních prostředí? Ano/Ne.
   - Z08K2O2 – Používají externí partneři organizace VDE pro přístup do interních prostředí? Ano/Ne.
   - Z08K2O3 – Používají externí partneři organizace virtuální API pro přístup do interních prostředí? Ano/Ne.

c) Z08K3 - Nedostupnost služeb
   - Pro měření může být zavedena i metrika pro měření času, který externista stráví v důsledku nedostupných služeb.

Z09 - Definovat metriky výkonnosti IT v organizaci a vyhodnocovat rychlost, kvalitu a efektivitu dodání softwaru od externího partnera

Tato zásada se týká především objednavatele, který musí měřit výkonnost IT za účelem zlepšování, hospodárného využívání zdrojů a odhalení odchylek od očekávání, pokud spolupracuje s externími partnery na vývoji softwaru. Zároveň musí vyhodnocovat rychlost, chybovost, kvalitu a spolehlivost dodání přírůstků či změn softwaru od externího partnera.

Vzhledem k tomu, že je velmi důležitá, je jí věnována samostatná kapitola *4.1.6 Měření výkonnost IT*. Když to shrnu, na metrikách záleží, neboť DevOps je postaven na kultuře neustálého zlepšování a metriky pomáhají prokázat toto zlepšení. V podmínkách externího vývoje softwaru jsou pak metriky kritickým prostředkem komunikace a vzájemné odpovědnosti mezi organizací (jejím IT oddělením a podnikáním) a outsourcingovým partnerem, který vyvíjí software. Je však potřeba dát na to, aby se neměřilo něco, co není důležité.

Dle mého názoru je nutné na jedné straně měřit návratnost investice na základě ukazatele ROI kvůli smysluplnosti investice do softwarového produktu či služby a na straně druhé je zásadní dle Puppet (2016) měřit výkonnost IT. Z hlediska průchodnosti kódu na základě metrik četnosti nasazování a doby potřebné k dodání změny a z hlediska stabilitu systémů na základě metrik střední doby zotavení a míry selhání změn.

Sadu metrik je možné doplnit o další běžné metriky v agilním vývoji softwaru, jako jsou: počet řádek kódu (anglicky Source lines of code – SLOC), pokrytí kódu testy, počet reportovaných chyb a rychlost vývoje v průběhu sprintu.

Velmi podstatné je ale požívat všechny metriky ve smyslu indikátorů na úrovni týmu a zaměřit se na příčiny. V žádném případě nesmí být podle nich hodnoceni lidé a dávat jim je do osobních cílů.

Význam: Nedílnou součástí partnerského vztahu s externím dodavatelem založeném na DevOps musí být měření výkonnosti IT. Co nelze měřit, nelze zlepšit. Důležité je to z důvodu toho, že je třeba nespoléhat se jen na subjektivní hodnocení v rámci zpětné vazby, ale i mít nějaké faktické údaje ve formě indikátorů, aby bylo možné dosáhnout neustálého zlepšování (anglicky Continuous improvement - CI). Rozhodnutí je třeba dělat jedině na základě výsledků a měřením. Měření by

mělo být prováděno na základě dodané funkcionality. A ne dle vykázaných MD vůči plánu. Lidé se často zaměřují jen na věci, které jsou vidět. Někdy ale je důležité měřit, co vidět není.

Hodnocení zásady *Z09 - Definovat metriky výkonnosti IT v organizaci a vyhodnocovat rychlost, kvalitu a efektivitu dodání softwaru od externího partnera* na základě následujících kritérií:

a) Z09K1 - Výkonnost IT
   Měření založené na odpovědích na otázky:
   - Z09K1O1 - Měří se návratnost investice? Ano/Ne.
   - Z09K1O2 - Měří se výkonnost IT podle metrik, které používají vysoce výkonné IT společnosti (Puppet)? Ano/Ne.
   - Z09K1O3 - Měří se výkonnost IT jiný způsobem?
   - Z09K1O4 - Používají se indikátory pro neustálé zlepšování nebo v rámci praktiky retrospektiva? Ano/Ne.

b) Z09K2 - Průchodnost kódu do produkce
   Pro měření je možné použít metriky průchodnosti kódu a porovnat je s vysoce výkonnými IT společnostmi (například na základě ročních reportů organizace Puppet):
   - Frekvence nasazování,
   - Doba potřebná k realizaci změny,
   - Počet chyb softwaru zjištěných v produkci.

c) Z09K3 - Stabilita systémů
   Pro měření je možné použít metriky stability systémů a porovnat je s vysoce výkonnými IT společnostmi (například na základě ročních reportů organizace Puppet):
   - Střední doba zotavení,
   - Míra selhání změn.

Z10 - Řídit a spravovat dodávku softwaru od externího dodavatele pomocí přístupů CI, CDE a CD[9]

Prostřednictvím DevOps je cílem zavést pro externí dodavatele procesy od průběžné integrace (anglicky Continuous Integration) přes testování až po nasazení do akceptačního prostředí, tj. průběžné dodávání (anglicky Continuous Delivery), nebo až po produkci, tj. plynulé nasazování (anglicky Continuous Deployment), které lze jednotně nastavit, podpořit nástroji a umožnit tak automatizaci, častější a kvalitnější výstupy a vyšší efektivitu. Nutné je ale konstatovat, že průběžné dodávání či plynulé nasazování závisí na typu aplikace a oboru působnosti zákazníka. Například v bance nelze mít plynulé nasazování kvůli bezpečnostním pravidlům a regulacím.

Význam: Úspěch DevOps a agilního způsobu vývoje závisí na přístupech CI a CDE nebo CD. Tyto přístupy lze podporovat různými nástroji. Následná automatizace procesů podporuje agilní vývoj a zaměření na sledování hodnoty pro zákazníka.

Hodnocení zásady *Z10 - Řídit a spravovat dodávku softwaru od externího dodavatele pomocí přístupů CI, CDE a CD* na základě následujících kritérií:

[9] Přístupy průběžné integrace, průběžného dodávání a plynulého nasazování jsou popsány v kapitole *2.5 Holistický přístup DevOps od vývoje po provoz softwaru.* Ve stejné kapitole byly zavedeny i zkratky CI, CDE a CD pro tyto přístupy.

a) Z10K1 – Uplatnění přístupů průběžné integrace (CI), průběžného dodávání (CDE) a plynulého nasazování (CD)
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z10K1O1 – Má organizace zavedený přístup průběžné integrace (CI)? Ano/Ne.
   - Z10K1O2 – Využívá organizace přístupu průběžného dodávání (CDE) pro dodávku softwaru? Ano/Ne.
   - Z10K1O3 - Využívá organizace přístupu plynulého nasazování (CD) pro dodávku softwaru? Ano/Ne.

b) Z10K2 - Firemní kultura pro přístupy CI, CDE a CD
   Měření pro toto kritérium je založené na odpovědích na otázky:
   - Z10K2O1 - Má organizace firemní kulturu, která umožňuje pracovat s pomocí přístupu CI? Ano/Ne.
   - Z10K2O2 - Má organizace firemní kulturu, která umožňuje pracovat v rámci přístupů CI a CDE? Ano/Ne.
   - Z10K2O3 - Má organizace firemní kulturu, která umožňuje pracovat v rámci přístupů CI a CD? Ano/Ne.

Z11 - Zavést, spravovat a udržovat důvěryhodnost a bezpečnost softwaru na straně objednavatele

V nastupujícím digitálním světě, ve kterém bude vývoj a provoz aplikací zajišťován především třetími stranami je nezbytná důvěryhodnost externího dodavatele a jeho softwaru. Z důvěryhodnosti dodavatele plyne i důvěryhodnost softwaru. Důvěryhodnost musí být zavedena, spravována a udržována v celém životním cyklu softwaru.

Detailněji se této zásadě věnuje kapitola *4.5 Důvěra v přístupu DevOps s integrovaným outsourcingem vývoje softwaru*.

Význam: Důvěryhodnost je nutné zavést, spravovat a udržovat v celém životním cyklu vývoje softwaru a po dobu trvání spolupráce s externím dodavatelem nebo poskytovatelem služeb. Uvědomit si závislosti na open-source kódu a kódu třetích stran. Je nutné snižovat rizika. Důležitý je integrovaný přístup.

Hodnocení zásady *Z11 - Zavést, spravovat a udržovat důvěryhodnost a bezpečnost softwaru na straně objednavatele* na základě následujících kritérií:

a) Z11K1 - Důvěryhodnost dodavatele
   Měření pro toto kritérium je založené:
   - Na ordinální bodovací stupnici v rozsahu škály od 0 do 5 bodů (posloupnost od nejmenší k největší). Hodnocení bude spočívat v udělování bodů, přičemž jednotlivé stupně škály jsou: 5 bodů - velmi vysoká důvěryhodnost; 4 body - vysoká důvěryhodnost; 3 body - střední důvěryhodnost; 2 body - nízká důvěryhodnost; 1 bod - velmi nízká důvěryhodnost; 0 - žádná.

b) Z11K2 - Důvěryhodnost softwaru
   Měření pro toto kritérium je založené:

- Na ordinální bodovací stupnici v rozsahu škály od 0 do 5 bodů (posloupnost od nejmenší k největší). Hodnocení bude spočívat v udělování bodů, přičemž jednotlivé stupně škály jsou: 5 bodů - velmi vysoká důvěryhodnost; 4 body - vysoká důvěryhodnost; 3 body - střední důvěryhodnost; 2 body - nízká důvěryhodnost; 1 bod - velmi nízká důvěryhodnost; 0 - žádná.

Z12 - Zavést controlling nákladů na straně objednavatele

Záměrem této zásady je zdůraznit, že v podmínkách DevOps a outsourcingu vývoje softwaru je na straně objednavatele důležitá schopnost řídit finance a náklady na základě nákladových metrik a controllingu.

Za prvé je controlling směrodatný z hlediska rozhodnutí o outsourcingu funkční oblasti. K převedení funkční oblasti vývoje softwaru na externího dodavatele přistupuje organizace zpravidla z prokazatelných ekonomických důvodů, tzn. jsou-li náklady na interní zajištění funkční oblasti vyšší, než je cena, kterou nabídne externí dodavatele. Při outsourcingu funkční oblasti jsou výhodou předvídatelné náklady a kontrola výdajů.

Za druhé je při agilním způsobu vývoje softwaru zásadní nejen spokojenost zákazníka, ale i finanční návratnost softwarového projektu. Z toho důvodu je nutné sledovat investiční náklady (anglicky non-recurrent) na vývoj softwaru a počítat ukazatel návratnosti investice ROI[10]. Zpravidla se ROI počítá na začátku projektu. Užitečné je ROI počítat znovu při ukončení vývoje softwaru a také i po jednom až dvou letech používání softwaru a zpětně ho otestovat (anglicky backtesting) na základě původního ROI. Investiční náklady se obvykle rozpočítávají (anuizace) na celé období užívání produktu nebo služby. Iterativní vývoj softwaru založený na sprintech, jako tomu například u agilní metodiky Scrum, pomáhá přesněji odhadovat náklady na celý vývoj softwaru. Vzhledem k tomu, že je vývoj rozdělen do jednotlivých sprintů, je možné brzy vyčíslit náklady na jeden sprint a tento odhad použít pro odhad nákladů celého produktu nebo služby.

Dále musí být v controllingu zahrnuta analýza průběžných provozních nákladů (anglicky recurrent), tj. kolik bude stát průběžně provoz softwaru, tj. pravidelné platby plynoucí ze servisních smluv (SLA), za administrátory v IT provozu apod.

Význam: Cílem je zajistit dostatečné financování projektů řízených agilně v podmínkách outsourcingu vývoje softwaru. Agilní vývojové týmy musí končit všechny své projekty nejen včas, nýbrž také v rámci plánovaného rozpočtu a v požadovaném rozsahu ke spokojenosti zákazníka. S pomocí controllingu nákladů lze udržet pod kontrolou jak investiční náklady, tak i vlastní provozní náklady nebo usilovat o jejich snížení. Controlling umožní také sledovat náklady na drobnější změny a opravy defektů a incidentů.

[10] Ukazatel ROI je popsán v kapitole 4.2.4 Měření výkonnosti IT.

Hodnocení zásady *Z12 - Zavést controlling nákladů na straně objednavatele* na základě následujících kritérií:

a) Z12K1 - Controlling nákladů v oblasti vývoje softwaru a jeho provozu
   Měření pro toto kritérium je založené na odpovědi na otázku:
   - Z12K1O1 - Má organizace zavedený controlling nákladů v oblasti vývoje softwaru a jeho provozu? Ano/Ne.
   - Z12K1O2 - Má organizace zavedený controlling nákladů pro agilní způsob vývoje softwaru? Ano/Ne.

4.2.2 Přínosy zavedení zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru

Tato kapitola má za cíl shrnout přínosy navržených zásad pro fungování DevOps a outsourcingu vývoje softwaru (očekávané přínosy), který bude organizace mít, pokud bude navržené zásady a pravidla uplatňovat a dodržovat. Tyto přínosy jsou důvodem formulace a aplikace zásad jako takových. Tato kapitola tak doplňuje naplnění dílčího cíle práce **DC4.**

Mezi přínosy zavedení zásad pro přístup DevOps ve spojení s realizací outsourcingu vývoje softwaru patří:

- Lepší způsob řešení tradičních problémů a zrušení informačních sil v důsledku zboření bariér mezi vývojem a IT provozem a zavedení principů DevOps
- Kvalitnější řešení softwarového produktu díky časté zpětné vazbě tvůrci softwaru
- Posílení komunikace a spolupráce díky DevOps
- Pravidelné dodávání hodnoty zákazníkovi v krátkých iteracích (sprintech)
- Rychlé vytváření produktů a tím zkrácení časového harmonogramu tvorby softwaru
- Rychlé nasazování, snížení chyb a zvýšení kvality s pomocí DevOps
- Dosažení stability IS a aplikací (spolupráce vývoje a provozu, rychlé obnovení při zhoršení úrovně služeb)
- Více času na inovativní práci v důsledku redukce ruční práce
- Rychlejší realizace uzavření smlouvy a zahájení vývoje softwaru (příprava)
- Spotřeba kvalifikovaných personálních zdrojů na straně externího dodavatele
- Flexibilita externích dodavatelů
- Využití technických znalostí a know-how externích dodavatelů
- Snadnější domáhání se náhrady škody v případě outsourcingu
- Snížení rizik při vývoji softwaru
- Zkrácení čekání na alokaci zdroje při jeho pokrytí externistou (odpadá dlouhá fáze výběru)
- Úspora nákladů[11] a času ve srovnání s vývojem softwaru vlastním silami
- Možnost vývoje softwaru distribuovanými týmy (a to i z více zemí)

[11] Softwarové společnosti jsou schopni vyvíjet a implementovat software za nižší náklady, neboť danou činnost zajišťují pro několik objednavatelů najednou, a využívají tak úspor z rozsahu.

4.2.1 Omezení, překážky a úskalí zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru

Tato kapitola je věnována omezením, překážkám a úskalím zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru. Obecně jsou zásady omezeny v tom, že se vztahují na střední až větší organizace.

Pro zásady byly identifikovány následující omezení, překážky a úskalí:

- Nedostatečná úroveň spolupráce lidí - DevOps a agilní přístupy vyžadují vysokou úroveň spolupráce a komunikace. U obou partnerů je nutné odstranit všechny překážky pro dobrou spolupráci.
- Tradiční kultura organizace - Kultura organizace obou partnerů musí nastavena na agilní přístupy a být v souladu s DevOps. Kultura organizace může být změněna postupně ve vlnách a to prostřednictvím pravidel, na nichž je kultura organizace postavena.
- Zneužívání postavení a síly - Ani jedna ze smluvních stran nesmí zneužívat svého postavení a síly.
- Rizika a kritické faktory outsourcingu - U zásad jsou totožná rizika a kritické faktory outsourcingu. Je nutné se snažit předvídat možné následky a předcházet jim.
- Experimentující vývojáři vs. stabilita provozu - Na jedné straně jsou experimentující vývojáři, na té druhé straně pak nutnost co nejstabilnějšího produkčního prostředí.
- Bezpečnost
- Rozmanitost organizací - Každá organizace je jiná. Co u jedné platí, u jiné nemusí platit.
- Smluvní zajištění agilního vývoje - Nevýhodou je poměrně zkomplikování smluv s externími poskytovateli služeb, neboť bude obsahovat mnoho dodatků a přídavných možností.
- Zdrojové kódy

4.2.2 Měření výkonnosti IT

Nedílnou součástí partnerského vztahu s externím dodavatelem založeném na DevOps musí být měření výkonnosti IT. Tato kapitola je věnována způsobům měření a definuje požadavky zásady *Z09 - Definovat metriky výkonnosti IT v organizaci a vyhodnocovat rychlost, kvalitu a efektivitu dodání softwaru od externího partnera* na měření výkonnosti IT. Bez měření není možné dosáhnout a udržet neustálé zlepšování a ani prokázat, jak funguje vývoj a dodávání softwaru od externího partnera. Po ruce je třeba mít fakta, aby mohla být zpětná vazba objektivní.

Měřit lze různými způsoby. Nástrojem měření jsou metriky a z nich vyplývající indikátory. Metriku můžeme definovat jako kritérium hodnotící vybrané charakteristiky určitého objektu například jeho kvalitu a efektivnost s pomocí různých hodnot. Objektem měření mohou být například proces, produkt či systém. Další informace poskytují indikátory, která vycházejí z metrik a jsou definovány na základě porovnávání výsledků metrik v čase nebo k jejich očekávanému výsledku. S jejich pomocí můžeme zjistit, zda se zlepšujeme, zhoršujeme nebo stagnujeme.

Z celkového hlediska tvorby a dodání softwarového produktu či služby lze obecně konstatovat na základě mé zkušenosti z mnoha projektů, že mezi základní metriky ve vztahu k nákladům

a výnosům patří ukazatel návratnosti investice ROI (z anglického Return On Investment). Vypočítává se odečtením odhadovaných nákladů od výnosů a následný vydělením rozdílu náklady. ROI udává míru návratnosti v procentech, tedy hodnotu zisku v procentech z investice, tj. utracených nákladů. Tento finanční ukazatel se používá pro hodnocení toho, zda se vložené prostředky jednou vrátí. Zkrátka se jím měří návratnost a efektivnost investic. Počítá se jak na začátku softwarového projektu, tak i na jeho závěr.

Pro měření výkonnosti IT se mohou použít metriky, které ve svých studiích používá Puppet (Puppet Labs, 2016), a které se zaměřuje na rovnováhu mezi vývojem a provozem. Výkonnost IT měří ve dvou hlavních kategoriích. Jednou je průchodnost kódu do produkčního prostředí a druhou je stabilita systému. Obě kategorie vzájemně souvisejí. DevOps prostředí pomáhá dosahovat rychlejší průchodnosti kódu a lepší stability systému, a to musí platit i v podmínkách externího vývoje softwaru. V každé z obou kategorií definuje Puppet dvě výkonnostní metriky (Puppet Labs, 2016).

Následujícími metrikami je možné měřit průchodnost kódu:

- Četnost nasazení (anglicky Deployment frequency)
  Měří, jak často organizace nasazuje kód, zda na vyžádání, což může být i vícekrát za den, nebo na týdenní či měsíční bázi.
- Doba potřebná k dodání změny (anglicky Change lead time)
  Měří čas od commitu kódu po úspěšně spuštěný kód v produkčním prostředí. Prakticky se jedná o rychlost nasazení, která ukazuje, jak rychle jsme schopni reagovat na chyby či změny a přizpůsobit se.

Následujícími metrikami je možné měřit stabilitu systémů:

- Střední doba zotavení (anglicky Mean time to recover - MTTR)
  Měří průměrnou dobu, jak rychle se měřená služba navrátí do běžného stavu či do posledního funkčního stavu v případě výskytu neočekávané události například neplánovaného výpadku, neúspěšného nasazení či zhoršení služeb. Pro obnovení služby je možná obnova ze zálohy databáze či aplikace, instalace balíčku předešlé verze či fixace chyby pomocí záplat (patchů). Vzhledem ke stabilitě systémů je třeba mít dobu pro obnovu co nejkratší.
- Míra selhání změn (anglicky Change failure rate)
  Měří procentní míru změn, které vedou buď ke zhoršeným službám, popřípadě výpadkům, nebo následně vyžadují nápravu, například formou hotfixu, obnovy ze zálohy či návratu k předešlé verzi. Tato míra by měla být co nejnižší. Pokud je odečtena od 100 %, výsledkem je míra úspěšnosti nasazení.

Výše uvedené metriky korelují se špičkovými prediktory, které používají vysoce výkonné IT společnosti, jako je například řízení verzí. Špičkové prediktory výkonnosti IT jsou popsány v kapitole *2.4 Principy DevOps*.

V podmínkách outsourcingu a DevOps je třeba všechny tyto klíčové ukazatele pravidelně sledovat a pracovat na jejich zlepšení.

Z hlediska sledování agilního vývoje softwaru patří mezi klasické metriky:

- Počet řádek kódu (anglicky Source lines of code – SLOC) je indikátorem kvality návrhu. Příliš vysoký počet řádek kódu znamená hodně práce s následnou údržbou. V takovém případě je v souladu s principem odstraňování odpadu potřeba, aby se vývojový tým zamyslel nad tím, jestli nelze něco udělat jednodušeji nebo případně smazat.
- Pokrytí kódu testy (anglicky Test coverage) je indikátorem spolehlivosti kódu a souvisí také s kvalitou návrhu. Udává, který kód takzvaně „stojí na vodě“, tedy nemohl být dobře testován a jeho funkčnost tak není zcela zaručena. V agilním vývoji však nelze usilovat o 100% pokrytí, to by bylo kontraproduktivní. Na druhou stranu při nízkém pokrytí bude další funkce stát více peněz.
- Počet reportovaných chyb (anglicky Reported bugs) je indikátorem kvality softwaru a rovnováhy mezi tvorbou softwaru a odchytáváním chyb. Chybám se nelze vyhnout, ale musí existovat nějaká míra, kterou je vývojový tým schopen řešit. Důležitý je nejen počet chyb, ale i jejich závažnost.
- Rychlost v agilním vývoji (anglicky Velocity) je indikátorem rychlosti týmu v průběhu sprintu. Pokud vývojový tým pokračuje příliš rychle, přidá Scrum Master další úkoly z Product Backlogu do Sprint Backlogu. Pokud je tým příliš pomalý, řeší Scrum Master důvody, proč se nedaří pracovat stanovenou rychlostí. Je třeba dbát na dlouhodobě udržitelné tempo a zákaz práce přesčas.

Se sledováním rychlosti v agilním vývoji souvisí Burn Down graf, který je klíčový pro znázornění zbývající práce a jak rychle se vyvíjí. Je z něj patrné, zda je potřeba upravit Project Backlog, nebo zda nastaly nějaké problémy.

Metriky umožňují organizaci a externímu partnerovi objektivně zhodnotit tempo a kvalitu práce, která je prováděná na produktech a službách v rámci projektů vývoje softwaru v podmínkách DevOps a agilních přístupů.

Při hodnocení metrik je důležité soustředit se na hledání příčiny a ne důsledku. V agilním vývoji je v žádném případě nelze implementovat jako tvrdé metriky, které budou použity jako cíle, na základě kterých budou lidé hodnoceni nebo které se dají do smlouvy s externím dodavatelem. Naopak by měly sloužit ve zpětné vazbě a jako ukazatelé trendů.

Ve vztahu k provozu aplikací jsou klíčovými naopak tvrdé metriky, které jsou definovány ve smlouvě SLA a patří k základním garantovaným parametrům softwarového produktu či služby. Podstatnými tvrdými metriky sjednanými pro sledování výkonnosti provozu aplikace a infrastruktury IT jsou především:

1. Dostupnost (anglicky Availability)
   V procentech udávaná schopnost softwarového produktu či služby provádět dohodnutou funkci v době, kdy je požadována. Vypočítává se jako procentuální podíl dohodnuté provozní doby služby a součtu časů výpadků. Dostupnost může být garantovaná jako sjednaný parametr služby a zjišťovaná jako kvalitativní ukazatel poskytované služby.
2. Doba odezvy (anglicky Response Time) má dva významy.
   a) Určuje čas dokončení operace nebo transakce v sekundách. Je podmíněna více faktory: přenosovou linkou, použitým hardwarem i architekturou aplikace. Zpravidla je

stanovena průměrná a mezní odezva aplikace v rámci služby. Uživatelé jsou na rychlost odezvy velmi citliví.

b) Určuje dobu zahájení řešení požadavků a incidentů v hodinách či dnech, které začíná okamžikem identifikace incidentu a končí zahájením diagnózy jeho příčin. Zpravidla jsou stanoveny různé lhůty reakcí v závislosti na prioritě incidentu.

Oba tyto ukazatele se pravidelně zjišťují a vyhodnocují, neboť se jimi kvantifikují rozdíly mezi sjednanými (očekávanými) parametry služby a službou, kterou zákazník skutečně dostane. Na jejich základě je možné předpovídat výkon procesů, což je důležité pro dodržení stanovených cílů úrovně služeb.

4.2.3 Zdrojové kódy softwaru

Předání zdrojových kódů softwaru by mělo být vždy upraveno smluvně například ve smlouvě o zhotovení a implementaci softwaru. V příslušném smluvním ujednání by mělo být řečeno několik věcí. Za prvé, že zdrojové kódy softwaru je nutné předat na konci každého sprintu – například ve fázi Vyhodnocení sprintu (anglicky Sprint Review nebo také Product demo). Za druhé povinnost, že objednatel obdrží i dokumentaci ke zdrojovým kódům. Buď ve formě srozumitelných komentářů, aby jim byl schopen někdo další rozumět, nebo v samostatné dokumentaci. Dokumentace by měla zahrnovat databázové modely, popis postupu vytvoření softwaru ze zdrojové formy a jeho instalaci, vysvětlení obsahu jednotlivých programových modulů a jejich klíčových funkcí ve formě komentářů ve zdrojových kódech. Toho není potřeba, jestliže vše obsahuje systémová analýza a návrh softwarového řešení, které jsou součástí smlouvy nebo byly předány objednavateli v prvních fázích.

Převzít zdrojové kódy je nutné z důvodu budoucího zajištění servisu a údržby softwaru.

Povinnost vydání zdrojových kódů existuje vždy, pokud se aplikuje na zhotovení softwaru právní úprava tzv. zaměstnaneckého díla, tj. uzavření smlouvy s programátorem (autorem).

4.3 Typy organizačních struktur týmů pro DevOps a outsourcing vývoje softwaru

DevOps vyžaduje čas, úsilí a organizační změny. Tato kapitola má za cíl navrhnout modely týmových organizačních struktur pro přístup DevOps v podmínkách outsourcingu vývoje či provozu softwaru na základě poznatků z kapitol 2., 3. a 4 a naplnit tak dílčí cíl práce **DC5.** Kapitola se tedy zabývá tím, jaké modely organizační týmové struktury jsou vhodné pro spolupráci s externím dodavatelem, aby bylo možné úspěšně spolupracovat a dosáhnout co možná nejlépe efektivního hodnotového toku.

Je zřejmé, že neexistuje žádné zázračné uspořádání nebo model týmové organizační struktury, který by vyhovoval každé organizaci. Nicméně je užitečné charakterizovat několik různých modelů pro týmové struktury, z nichž některé mohou vyhovovat organizacím lépe než jiné. Cílem je reorganizace týmů, aby mohli poskytnout hodnotu zákazníkům.

Kim (Kim et al. 2016, s. 77) říká, že strukturu organizace a architekturu je třeba stavět s ohledem na Conwayův zákon (anglicky Conway's Law). Ten uvádí, že organizace, které konstruují systémy, jsou nuceny produkovat návrhy, které jsou kopiemi jejich komunikačních struktur. Z toho vyplývá konstatování, že to, jak jsou organizovány týmy, ovlivňuje, jakým způsobem provádí svou práci.

V klasickém manažerském pojetí existují následující typy organizací:

- Funkčně orientované organizace mají hierarchické organizační struktury a odborné znalosti jsou centralizovány, což umožňuje profesní růst a rozvoj dovedností.
- Maticově orientované organizace se snaží spojit funkční a tržní orientaci. V této poměrně komplikované organizační struktuře spočívá nebezpečí v tom, že by jednotlivec mohl reportovat dvěma manažerům.
- Tržně orientované organizace jsou orientovány tak, aby mohly rychle reagovat na potřeby zákazníků. Tyto organizace mají tendenci mít plochou organizační strukturu a skládají se z několika křížových funkčních disciplín (marketing, inženýrství, prodeje a tak dále), které často vedou k potenciálnímu nadstavu zaměstnanců a tím k nadbytečnosti zdrojů v celé organizaci.

Organizační strukturu celé organizace je třeba reflektovat při návrhu modelů typů organizační struktury pro vývoj a provoz softwaru.

Základem DevOps, jak už vyplývá z principu multidisciplinárních autonomních týmů, je tvořit odpovědné multidisciplinární týmy (anglicky cross-functional), které budou zodpovědné za konkrétní služby nebo aplikace a dodání softwarových produktů. Tímto způsobem mohou být iniciativní v rozvoji a řešení problémů.

Na základě zkušeností autora této práce z mnoha let praxe především v oblasti IT provozu, informačních zdrojů o DevOps a veřejného reportu od organizace Puppet (Puppet Labs, 2018) byly v této práci navrženy a charakterizovány následující typy organizačních struktur, které mohou být více či méně použity ve spolupráci s externím dodavatelem pro realizaci outsourcingu vývoje softwaru. Z tohoto důvodu byly setříděny podle toho, jaké typ organizační struktury se jevil jako nejvhodnější z hlediska outsourcingu vývoje softwaru.

1. Spojený tým IT pro DevOps spolupráci
2. Multidisciplinární týmy pro specifické služby nebo aplikace
3. Provoz s flexibilní infrastrukturou (Infrastructure-as-a-Service)
4. Technický tým SRE (model Google)
5. DevOps schopnosti jako služba od externích poskytovatelů (malá firma)
6. Dedikovaný tým DevOps

Do budoucna je možné počítat s další evolucí DevOps, které se projeví v tom, že se jistě objeví další modely.

Je možné konstatovat, že v rámci této práce byly navrženy modely číslo 1, 3, 5 a 6. Důležité přitom bylo, zda jsou jednotlivé modely reálně uplatnitelné, proto byl autorem této práce proveden samostatný průzkum jejich využívání. Jak se ukázalo z posledních výzkumů organizace Puppet (Puppet Labs, 2018), podobné modely se používají v praxi. Žádné podrobné informace k nim ale

nejsou dostupné. Proto charakteristika těchto modelů týmových organizačních struktur je jedním z hlavních přínosů této práce.

Do výše uvedeného seznamu byly přidány dva modely. Model č. 2: Multidisciplinární týmy pro specifické služby nebo aplikace, který představuje preferovaný způsob organizace týmů DevOps podle DevOps Agile Skills Association (2017, s. 81) a je představen v kapitole *2.6 Týmová organizační struktura pro DevOps*. Druhý přidaný model je číslo 4: Technický tým SRE (model Google), který organizace Puppet (Puppet Labs, 2018) uvádí jako efektivní model pro společnosti s jedním softwarovým produktem a proto byl také zařazen do seznamu.

V této souvislosti je třeba upozornit na zajímavou informaci od organizace Puppet (Puppet Labs, 2018), že dvě nejpoužívanější organizační struktury pro DevOps jsou „Multidisciplinární týmy pro specifické služby nebo aplikace" a „centralizované IT týmy", za nimiž následuje „oddaný tým DevOps".

4.3.1 Spojený tým IT pro DevOps spolupráci

Pro hladkou spolupráci mezi vývojovými týmy a provozními týmy je nutné mít jasně vyjádřený společný cíl. Příkladem takového cíle může být: Dodávání častých a spolehlivých změn. Každý z týmů se specializuje v tom, co je nutné zajistit, ale také sdílejí společně to, co je potřeba. Tento model vyžaduje organizační změny. Odborníci IT provozu musejí být smícháni s vývojáři, přičemž může existovat více oddělených vývojových týmů, z nichž každý může pracovat na nějaké aplikaci či službě. Pro efektivní spolupráci musejí být obeznámeni s praktikami vývojářů a jejich nástroji, například s řízením verzí (Git, TFS, SVN) a metodikami testování. Vývojové týmy na oplátku musejí brát vážně problematiku provozu. Všichni mohou být soustředěni a sedět v jednom společném prostoru. To posiluje spolupráci a komunikaci. Z toho zřejmé je soustředění vývojářů a odborníů IT provozu do jednoho IT týmu. Tento model je vysoce účinný z hlediska zvyšování výkonnosti IT a spolupráce a může být předstupněm k typu modelu *Multidisciplinárních týmů zodpovědných za konkrétní služby nebo aplikace*. Modelu spojeného IT týmu pro DevOps spolupráci mohou využívat technologické organizace a banky, které chtějí využít potenciálu spolupráce DevOps. Autor této práce se setkal s tímto modelem týmové organizační strukturou spojeného IT týmu v bance, kde pracuje.

Tento typ modelu je možné použít pro spolupráci s externími dodavateli. Externí vývojové týmy či jednotliví externí by měli být umístěni lokálně v organizaci objednavatele, aby mohly úzce spolupracovat s jeho IT odborníky.

Model spojeného IT týmu pro DevOps spolupráci může mít i formu plně sdílené odpovědnosti za provoz. Pokud má organizace buď jeden nebo jen několik softwarových produktů či služeb, je vysoce efektivní a osvědčené integrovat provoz do odpovědnosti a práce vývojových týmů. Všichni se tak zaměřují na společný účel. Plně sdílenou odpovědnost za provoz využívají společnosti jako Netflix a Facebook, které uspěly s jediným webovým produktem. Není však uplatnitelná u organizací s širokou paletou produktů a služeb. Tato modifikace modelu spojeného IT týmu však znemožňuje realizaci outsourcingu vývoje softwaru. Lze si ale představit, že si společnost najme na výpomoc jednotlivé externí vývojáře jako živnostníky. Hlavní odpovědnost a řízení vývoje softwaru zůstává v kompetenci vlastních zaměstnanců společnosti.

4.3.2 Multidisciplinární týmy

Jeden z principů DevOps říká, že je nutné mít multidisciplinární autonomní týmy pro tvorbu softwarových produktů a jejich provoz. Takové týmy mohou být označeny jako Business system týmy, které jsou podporované jedním či více týmy Platforem. Business system týmy jsou zodpovědné za konkrétní aplikace nebo služby. Business system týmy požadují a používají produkty a nástroje, které vytvářejí a provozují týmy Platforem. Model Business system týmů s podporou týmů Platformy charakterizuje DevOps Agile Skills Association (2017, s. 81) a je popsán v kapitole *2.6 Typy týmových organizačních struktur pro DevOps*. Jiné informační zdroje o DevOps, jako například Kim (Kim et al. 2016), popisují potřebu multidisciplinární autonomních týmů, ale výslovně týmy vývoje a provozu nedělí na Business system týmy a týmy Platformy.

Tento typ model je možné použít pro spolupráci s externími dodavateli při částečně outsourcovaném vývoji softwaru. Business systém tým může být vytvořen ze zaměstnanců externího dodavatele a poskytnout organizaci objednavatele. Rovněž některé komponenty platforem mohou být provozovány ze strany externích poskytovatelů.

4.3.1 Technický tým SRE (model Google)

Některé organizace (včetně společnosti Google) používají jiný model, jak uvádí Puppet (Puppet Labs, 2018), s přímou předávkou softwarových produktů a služeb od vývojového týmu do týmu zajišťujícího odpovídající úroveň spolehlivosti systémů, služeb a produktů (anglicky Site Reliability Engineering - SRE). Tým SRE tedy přebírá software a provozuje ho. V tomto modelu musí vývojový tým poskytnout důkazy o testování (logy, metriky atd.) týmu SRE, které ukazují, že jejich software je dostatečně kvalitní, aby byl podporován týmem SRE.

Rozhodující je, že tým SRE může odmítnout software, který je nedostatečně způsobilý pro provoz, a požádat vývojáře o zdokonalení kódu ještě před tím, než bude nasazen do produkce. Spolupráce mezi vývojáři a týmem SRE probíhá na základě provozních kritérií. Jakmile je ale tým SRE spokojen s kódem, je tímto týmem podporován v produkci (a nikoli vývojovým týmem). Tento model používají organizace s vysokou mírou technické a organizační zralosti.

Tento typ modelu je možné použít pro spolupráci s externími dodavateli při částečně outsoursovaném vývoji softwaru s tím, že tým SRE bude přebírat softwarové produkty a služby od externího dodavatele a následně je provozovat.

Tento typ modelu je určen pro organizace, které sice provozují své aplikace ve veřejném cloudu například Microsoft Azure nebo Amazon EC2, ale vývoj softwaru si zachovávají ve své hlavní činnosti. Je zřejmé, že velká část provozního týmu není potřeba. Na týmu externího poskytovatele je, aby organizaci poskytoval na vyžádání službu infrastruktury (IaaS). Tento externí tým přebírá větší část zodpovědnosti za provoz a pružně poskytuje infrastrukturu, na které jsou nasazovány a provozovány aplikace. Vlastní provozní tým organizace je tedy na úrovni externího poskytovatele.

Vedle toho vývojový tým musí mít odborné znalosti o rámcích pro řízení a provozu IT služeb (ITIL), metrikách, monitorování, správě serverů a tak dále, aby mohl nad rámec svých tradičních úkolů a postupů zajišťovat komunikaci s externím servisním týmem cloudového poskytovatele (IaaS). Výhodou je snadná implementace toho modelu. Na druhou stranu se v něm ztrácí přímá

spolupráce s odborníky z provozu a to může ohrozit přechod na efektivnější typy modelů, o který by organizace měla usilovat.

Možným rozšířením IaaS je pak PaaS (Platform-as-a-Service), kde externí provozní tým poskytuje pružně úplné výpočetní prostředí.

Tento typ modelu rovněž předpokládá interní vývojový tým, který bude řídit jak případné externí vývojáře, které si může najmout na výpomoc, tak i servisní tým cloudového poskytovatele. V tomto modelu je možné jen velmi omezeně outsourcovat vývoj softwaru.

4.3.2 DevOps schopnosti jako služba od externích poskytovatelů (malá firma)

Nelze opomenout malé firmy, které mohou také vytvářet softwarové produkty či služby a mít potřebu je provozovat. Tyto firmy ale nemají zpravidla dostatek financí, zkušeností nebo nemají zaměstnance s odpovídající kvalifikací, aby zajistily provoz ve všech jeho aspektech. Vývojový tým může oslovit specializovaného externího poskytovatele, aby jim pomohl vytvářet testovací prostředí, spravovat konfiguraci, automatizovat jejich infrastrukturu a monitorování a poradil jim s integrací provozních aspektů do jejich vývoje softwaru. Tímto pragmatickým způsobem se to vše může naučit a s růstem firmy a počtu zaměstnanců se následně postupně přesunout na typ modelu *Centralizovaného IT týmu pro DevOps spolupráci* nebo typu modelu *Multidisciplinárních týmů zodpovědných za konkrétní služby nebo aplikace*.

Tento typ modelu je možné použít pro outsourcing softwaru na externí dodavatele. U menších firem. Protože se ale předpokládá, že tvorba softwaru patří mezi hlavní činnosti firmy, není vhodné uvažovat o realizaci outsourcingu vývoje softwaru. Firma si ale může v tomto modelu najmout na výpomoc externí vývojáře jako živnostníky. Hlavní odpovědnost a řízení vývoje softwaru zůstává v kompetenci vlastních zaměstnanců firmy.

O modelu, ve kterém by menší firma, neměla vývojáře a ani IT odborníky na provoz a chtěla si outsourcovat vývoj softwaru ani neuvažuji, protože by to bylo složité a dlouhodobě by to nemohlo nefungovat. Nebylo by koho učit o DevOps na straně firmy a tudíž by byl vyloučen přechod na některý z vyšších modelů týmových organizačních struktur. V takovém případě je finančně efektivnější si pořídit krabicové řešení a customizovat ho na potřeby firmy, aby se mohla soustředit na svoji hlavní výdělečnou činnost.

4.3.3 Dedikovaný tým DevOps

Dedikovaný tým DevOps je možné vytvořit pro zavedení přístupu DevOps v organizaci. Výzkumy Puppet (Puppet Labs, 2018) ukazují, že tento typ modelu organizace skutečně využívají. Měl by ale mít jen dočasný charakter s jasným mandátem přiblížit k sobě Dev a Ops. Dle mého názoru by měl trvat méně než 12 nebo 18 měsíců. Tento typ modelu je vhodný jako přechodný do organizací, které mají velký rozdíl mezi vývojem a provozem nebo kde k tomu existují silné tendence.

Na starosti má zavedení praktik agilních přístupů, procesů CD a automatizace pro odborníky provozu a pro vývojové týmy zas vysvětlení problematiky stability, provozu a údržby systémů a aplikací jako například rozdělovače zátěže (anglicky load-balancers), řízení kontinuity činností organizace (BCM) a tak dále. Jeho hlavním cílem je šířit povědomí o DevOps, trvalá změna myšlení a kultury organizace. Pokud lidé uvidí reálné přínosy úzké spolupráce vývojářů a odborníků na provoz, má reálnou šanci dosáhnout svého cíle. Hlavním úkolem tohoto týmu je přiblížit se na

úroveň typu modelu *Centralizovaného IT týmu pro DevOps spolupráci* nebo typu modelu *Multidisciplinárních týmů zodpovědných za konkrétní služby nebo aplikace*. Na druhou stranu je třeba dbát na to, aby se z dedikovaného DevOps týmu nestalo informační silo, což by bylo kontraproduktivní. Odpovědnost za nasazování a monitoring produkčních systémů mají nadále provozní týmy.

V případě externího dodavatele je potřeba, aby ovládal DevOps schopnosti a neplýtvalo se penězi na zřízení dedikovaného týmu DevOps. Z toho vyplývá, že tento typ modelu není zcela nejvhodnější pro spolupráci s externím dodavatelem.

4.4 Důvěryhodnost partnerů poskytujících outsourcing vývoje softwaru

V DevOps světě se hranice životní cyklu vývoje softwaru stávají méně zřetelné a rozšiřují se do provozu.

Při vývoji softwaru nezačínají vývojáři pracovat od nuly. Naopak stále častěji opakovaně využívají stávající softwarové komponenty, komponenty s otevřeným zdrojovým kódem (anglicky open-source software - OSS) a vzdálená rozhraní API třetích stran, u nichž nemusí být výjimečné, že také využívají dalších API rozhraní. Do procesu vývoje softwaru je zapojeno množství aktérů, z nichž většina je externími subjekty, tj. dodavatelé, poskytovatelé služeb, partneři, klienti, a vznikají závislosti na komponentách třetích stran. Kromě toho toto vše pak může jako službu dodávat softwarová společnost, která může mít také své externí dodavatele (subdodávky). Jedním z hlavních důvodů je značně rozšířené používaní softwaru s otevřeným zdrojovým kódem (OSS), který poskytuje softwarová aktiva v rozsahu a složitosti od malých kusů kódů, modulů, knihoven, rámců a aplikací až po celé softwarové systémy jako je Linux, a virtualizačních technologií, jako jsou například virtuální servery a kontejnery.

4.4.1 Zavést, spravovat a udržovat důvěru a bezpečnost

Tyto skutečnosti mě vedou k tomu, že považuji za důležité vědomě zavést, spravovat a udržovat důvěru a bezpečnost ve výše uvedených závislostech a mezi všemi aktéry a nenechat se jimi ohrozit. Důvěryhodnost externího dodavatele a jeho softwaru je zcela základní nutností a je nezbytnou podmínkou pro dodávku a provoz funkčního software. Z toho vyplývá, že musí být integrována do celého životního cyklu vývoje softwaru.

Důvěra v software spočívá dle Gartner (2017) na třech základních atributech bezpečnosti softwaru, kterými jsou integrita, důvěrnost a dostupnost softwaru.

Důvěrnost souvisí s tím, že citlivé údaje mohou být předané třetí straně prostřednictvím API rozhraní. Proto musí být zajištěno to, aby přístup k důvěrným údajům byl omezen a tím se snížilo riziko jejich úniku.

Integrita softwaru určuje, že software musí být nezměněný, celistvý a být to, co říká, že je.

Dostupnost se vztahuje na to, aby byl software funkční a data dostupná uživateli v prostředí s integrovanými externími systémy, jako jsou cloudové služby (infrastruktura jako služba, platforma

jako služba a další) od externích poskytovatelů, a externí API rozhraní. Zajištěno musí být to, aby výpadek API rozhraní neměl negativní dopady na závislé aplikace.

Důvěryhodnost lze klasifikovat v několika úrovních. Gartner (2017) definuje celkem šest úrovní důvěryhodnosti, které by mohly být použity pro měření a informování o úrovni spolehlivosti a rizika při použití kódu nebo API třetích stran. Koncept šesti úrovní důvěryhodnosti znázorňuje následující tabulka:

Tabulka 4.1 – Klasifikace důvěryhodnosti

Vrstva	**Úroveň**	**Popis**
Důvěryhodná	Zaručená	Úplně spolehlivá s téměř nulovým rizikem, tj. bez známých obav
Částečně důvěryhodná	Potvrzená	Spolehlivý zdroj, ale dosud zcela bez rizika, tj. nízké riziko
Částečně důvěryhodná	Prokázaná	Důkaz o spolehlivosti se podaří čas od času, ale buď není konzistentní nebo není dostatečně spolehlivá, tj. střední riziko
Částečně důvěryhodná	Uznaná	Časné znamení důvěry a spolehlivosti potvrzeny, ale nejsou zcela prokázané, tj. střední riziko
Částečně důvěryhodná	Potvrzená	Předpokládaná důvěra v kód, ale nesou uznané žádné prokázané důkazní body, tj. vysoké riziko
Nedůvěryhodná	Neznámé	Nejvyšší riziko, neboť není žádné hodnocení důvěry nebo je neznámý negativního výsledku. Varování.

Zdroj: vlastní úprava dle (Gartner, 2017).

Gartner (2017) navrhuje zavést hodnocení důvěryhodnosti do životního cyklu vývoje softwaru. Zajišťování důvěryhodnosti softwaru musí být součástí práce vývojářů.

Hodnocení důvěryhodnosti ale něco stojí. Je třeba zvážit, v jakých situacích se vyplatí.

4.4.2 Opatření pro zajištění důvěryhodnosti

Z hlediska opatření pro ověření důvěryhodnosti externího dodavatele je třeba zjistit jeho majetkové i organizační poměry z nezávislých zdrojů například z registrů. Externí dodavatel musí být majetkově i personálně oddělená firma od konkurentů. Rovněž je třeba dbát na to, že nespolupracuje s konkurencí a to i v budoucnu, aby k ní neunikalo cenné know-how a důvěrné informace, které by mohla zneužít pro získání konkurenční výhody. S dodavatelem jakožto příjemcem důvěrných informací musí být uzavřená dohoda o mlčenlivosti.

Pro externí komponenty a API rozhraní třetích stran, které využívají vývojáři nebo software, musí být nejprve zajištěna evidence v katalozích či repozitářích a následně musí být posouzena jejich úroveň důvěryhodnosti. Pro externí komponenty včetně těch s otevřeným zdrojovým kódem lze využít nějakého SCA nástroje, který umožní analyzovat složení aplikací za účelem zjištění komponent, o kterých je známo, že mají bezpečnostní či funkční zranitelnosti nebo vyžadují mít řádnou licenci. U rozhraní API se musí navíc sledovat závislosti. Rovněž je nutné na komponenty třetích stran použít zásady vnitřní bezpečnosti. Například zavést proces k upozornění, pokud bude zjištěna bezpečnostní chyba v zabezpečení softwaru, nebo pokud budou uvolněny nové verze. Veškerá externí API rozhraní, které se používají, je nutné nepřetržitě proaktivně monitorovat zejména z hlediska dostupnosti.

5 Ověření navrženého souboru zásad pro DevOps v podmínkách outsourcingu vývoje softwaru

Tato kapitola má za cíl ověřit zásady pro přístup DevOps v podmínkách outsourcingu vývoje softwaru, které byly navržené v kapitole 4, v softwarovém projektu na vytvoření aplikace pro tvorbu smluvní dokumentace a naplnit tak dílčí cíl práce **DC6**. Způsob ověření zásad je popsán v kapitole 4.1 *Hodnocení zásad*. Jedná se o navržený postup hodnocení pro všechny zásady, který může probíhat iterativně. Základem ověření zásad je vyhodnocení kritérií, která byla definována pro jednotlivé zásady v kapitole 4.2.

5.1 Ověření navržených zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru

Tato kapitola je věnována uplatnění zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru a jejich ověření v praxi v softwarovém projektu vývoje aplikace pro tvorbu smluvní dokumentace v nejmenované bance (objednavatel). Vývoj aplikace je zajišťován externím dodavatelem. Pro účely této práce pojmenujme aplikaci jako Generátor dokumentů.

V následujících podkapitolách je posuzováno splnění všech navržených zásad na základě hodnocení kritérií definovaných pro jednotlivé zásady. Výsledky měření a hodnocení splnění zásad byly zaznamenány v tabulkách kritérií jednotlivých zásad. Pro jednoznačnou identifikaci kritérií obsahuje titulek každé tabulky jméno kritéria. Vstupní informace a data pro hodnocení byly získány za prvé metodami pozorováním a dotazováním se členů vývojového a testovacího týmu, zástupců byznysu a projektového manažera. Za druhé z výsledků DRP testů nebo za třetí z incident managementu a nástroje JIRA, ke kterým má autor práce autorizovaný přístup.

Z přístupu a kultury DevOps se v organizaci objednavatele prosazuje jen několik principů. Pro vytvoření DevOps organizace byla provedena změna jen částečná v oblasti organizační. Odpovědnost za vývoj a provoz byla soustředěna na jednom místě v rámci jednotlivých domén IT. Zahrnuta je v ní i odpovědnost za náklady řešení, jak vývoje aplikací, tak i jejich provozu, a odpovědnost/pravomoc v oblasti aplikačních roadmap. Analytici, vývojáři, aplikační inženýři a správci sedí v kanceláři pohromadě spolu se svými týmovými vedoucími a IT manažerem domény.

Původním záměrem změny organizace bylo i zvýšit možnosti průběžného dodávání (anglicky Continuous Delivery), rozšířit používání automatizovaného nasazování (např. Urban Code) a snížit závislosti mezi systémy a aplikacemi při velkých aplikačních releasech. Ani jedna z těchto věcí se však neprosadila v IT doméně banky, kde pracuje autor této práce.

Z celkového hlediska se IT profiluje jako zprostředkovatel služeb (anglicky service broker), který má za cíl zajistit optimální mix na jedné straně mezi vývojem řešení na míru a nákupem balíkových řešení a na druhé straně mezi vlastními experty, odborníky různých profesí najatými přes agentury nebo softwarové společnosti (tzv. bodyshop) a externími dodavateli.

5.1.1 Z01 - Zajistit si infrastrukturu jako službu - virtualizace a cloud computing

1. Zjištění, ověření a posouzení současného stavu.
   Infrastruktura se průběžně optimalizuje z hlediska nákladů. Vzhledem k pokročilé virtualizaci se úspory realizují především v oblasti licencí. S objednáváním infrastruktury, především pak serverů, je spojeno hodně administrativy a komunikace. Na případné posilování parametrů serverů a stejně tak na přidávání prostředí či jejich rozšiřování je třeba myslet ve velkém předstihu při plánování ročního finančního rozpočtu.
2. Vyhodnocení kritérií této zásady:
   a) Z01K1 - Zavádění a využívání virtualizačních a cloudových technologií.
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

   Tabulka 5.1 – Z01K1 - Zavádění a využívání virtualizačních a cloudových technologií

Kritérium	Otázka	Vyhod.
Z01K1O1	Využívá organizace virtualizaci infrastruktury (serverů, úložišť dat)?	Ano
Z01K1O2	Využívá organizace virtualizaci sítí?	Ne
Z01K1O3	Využívá organizace virtualizaci služeb?	Ano
Z01K1O4	Využívá organizace virtualizaci desktopů?	Ano
Z01K1O5	Má organizace zavedené služby typu IaaS?	Ano
Z01K1O6	Má organizace zavedené služby typu PaaS pro vývojové týmy?	Ne
Z01K1O7	Využívá organizace služby typu SaaS?	Ano

Zdroj: vlastní zpracování

   b) Z01K2 - Poskytování prostředků infrastruktury pomocí samoobslužných služeb.
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

   Tabulka 5.2 – Z01K2 - Poskytování prostředků infrastruktury pomocí samoobslužných služeb

Kritérium	Otázka	Vyhod.
Z01K2O1	Jsou vývojovým týmům v organizaci poskytovány prostředky infrastruktury pomocí samoobslužných služeb?	Ne

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Standardizovat platformy, vytvořit balíčky virtualizované infrastruktury a nabídnout je prostřednictvím samoobslužných služeb multidisciplinárním (vývojovým) týmům.
   - Umožnit multidisciplinárním týmům na kliknutí dle potřeby objednávání a konfiguraci virtualizovaných balíčků infrastruktury pomocí samoobslužných služeb.

5.1.2 Z02 - Využít maximálně principů a procesů DevOps na základě agilních přístupů

1. Zjištění, ověření a posouzení současného stavu: V následujících odstavcích je uvedeno několik bodů. Komplexní zhodnocení úrovně DevOps a agilního vývoje softwaru v organizaci jako je banka je na samostatnou práci.

Vývoj softwaru (včetně analýzy)

V některých případech se nepodařilo odlišit důležité požadavky od těch méně významných, protože na tom uživatelé trvali. To se odrazilo v prioritizaci úkolů pro vývojáře, kteří se museli soustředit na méně významné požadavky na místo toho, aby řešili důležité věci, které přinášely největší hodnotu zákazníkovi.
Tvorbu softwaru zajišťují externí vývojáři nástroji a infrastruktuře objednavatele.

Testování
Pro testování ve vývojovém prostředí se využívá metodika Data-driven testování. V integračním a předprodukčním prostředí však testují testeři a uživatelé ručně a výsledky rovněž ručně zaznamenávají do JIRA.
Metodika testování Data-driven umožňuje zautomatizovat celý proces dodávky nové verze a zavést Plynulé nasazování (anglicky Continuous deployment).

Nasazování
Nasazování instalačního balíčku probíhá ručně. Do integračního prostředí nasazují vývojáři a do předprodukčního prostředí nasazuje aplikační správce dle pokynů uvedených v instalačním manuálu.

Provoz
Provoz zajišťují interní ÏT specialisti objednavatele. Dokumentace je poskytována česky v elektrické formě ze strany externího dodavatele.

2. Vyhodnocení kritérií této zásady:
   a) Z02K1 - Soulad s procesy DevOps a agilními přístupy.
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.3 – Z02K1 - Soulad s procesy DevOps a agilními přístupy

Kritérium	Otázka	Vyhod.
Z02K1O1	Má dodavatel prokazatelnou zkušenost s principy a procesy DevOps?	Ano
Z02K1O2	Je proces vývoje (včetně analýzy) v souladu s DevOps a agilními přístupy a principy?	Ne
Z02K1O3	Je proces integrace v souladu s DevOps a agilními přístupy a principy?	Ano
Z02K1O4	Je testování v souladu s DevOps a agilními přístupy a principy?	Ne
Z02K1O5	Je proces nasazení v souladu s DevOps?	Ne
Z02K1O6	Jsou procesy provozu a údržby v souladu s DevOps?	Ne

Zdroj: vlastní zpracování

   b) Z02K2 - Spokojenost s výsledným softwarovým produktem nebo s přírůstky dodanými na konci sprintů ze strany objednavatele a ze strany zákazníků.
   Výsledky měření pro toto kritérium, které je založené na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů, je uvedené v následující tabulce:

Tabulka 5.4 – Z02K2 - Spokojenost s výsledným softwarovým produktem nebo s přírůstky dodanými na konci sprintů ze strany objednavatele a ze strany zákazníků.

	Body	Míra pociťovaného uspokojení
Z02K2	6	Spokojen mírně nadprůměrně.

Zdroj: vlastní zpracování

c) Z02K3 – Kvalita softwaru
Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.5 – Z02K3 -Kvalita softwaru

Metrika kvality softwaru	Softwarový projekt
Počet reportovaných chyb	10

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Zavést agilní metodiky v softwarových projektech.
   - Zavést jednoho dedikovaného Vlastníka produktu (Product owner) pro lepší prioritizaci Back logu, který bude skutečným arbitrem a postará se o tom, aby se důležité věci, které přinášejí hodnotu, dělali dříve a ty méně důležité až později.
   - Držet se u požadavků Paretova pravidla 80/20. Toto pravidlo se používá u úspěšně dodaných softwarových produktů a říká, že 20 % funkcionality obsahuje 80 % byznys hodnoty a naopak zbylých 80 % funkcionality poskytuje jen 20% byznys hodnoty, neboť velká část funkcionality se buď vůbec nepoužije, nebo ji zřídka použije jen minimum uživatelů.
   - Automatizovat testování (velký potenciál automatizace)
   - Automatizovat nasazování (velký potenciál automatizace)
   - Zvážit zavedení a nastavení tzv. Plynulého nasazování (anglicky Continuous deployment).
   - Zvážit realizaci outsourcingu také pro funkční oblast Provozu a údržby softwaru. Smluvní záruky pro provoz jsou obsaženy v parametrech SLA, které definuje byznys.

5.1.3 Z03 - Usnadnit zahájení a ukončení spolupráce pro oba partnery

1. Zjištění, ověření a posouzení současného stavu:
   Smlouvy s externími dodavateli jsou víceméně standardizované. Zde se mi nepodařilo získat konkrétní znění smluv, neboť smlouvy s dodavateli jsou důvěrné. Nicméně platí, že smlouvy mají standardní ráz a nejsou nastaveny na agilní vývoj. Vývojové týmy, které chtějí vyvíjet agilně ve spolupráci s externími dodavateli, musí vyvinout značné úsilí na vyjednávání o smlouvě ve spolupráci s útvarem Nákupu a s právníky.
2. Vyhodnocení kritérií:
   a) Z03K1 - Odstoupení od smlouvy a ukončení spolupráce
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.6 – Z03K1 - Odstoupení od smlouvy a ukončení spolupráce

Kritérium	Otázka	Vyhod.
Z03K1O1	V případě nekvalitního dodavatele je možné snadno odstoupit od smlouvy na vývoj softwaru na konci iterace (sprintu)?	Ne
Z03K1O2	Jsou smluvní ustanovení ohledně odstoupení od smlouvy precizně formulovaná?	Ne

Zdroj: vlastní zpracování

b) Z03K2 - Schopnost identifikovat okamžik dodání softwaru
Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.7 – Z03K2 - Schopnost identifikovat okamžik dodání softwaru

Kritérium	Otázka	Vyhod.
Z03K2O1	Sleduje organizace naplnění podstatných požadavků s hodnotou pro zákazníka (například vůči business case v RFI)?	Ne
Z03K2O2	V případě naplnění podstatných požadavků s hodnotou pro zákazníka (splnění business case) schopen objednavatel převzít dodávku softwaru a rozvíjet ho bez externího dodavatele?	Ne

Zdroj: vlastní zpracování

Výměna dodavatele není snadná.

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Přeformulovat smlouvy vzhledem k přechodu na agilní vývoj.
   - Precizně formulovat smluvní ustanovení ohledně odstoupení od smlouvy pro případ odvedení nekvalitní práce nebo žádné práce.
   - Sledovat a prioritizovat naplnění podstatných požadavků s hodnotou pro zákazníka (pro splnění business case).

5.1.4 Z04 - Zkvalitnit spolupráci a rozsah výměny informací mezi partnery

1. Zjištění, ověření a posouzení současného stavu:
   Vývojový tým, který pracuje na projektu, je relativně malý. V rámci týmu je spolupráce na dobré úrovni. Členové týmu si dobře sedli. Velmi efektně se domlouvají na společném postupu a plnění dílčích úkolů.
2. Vyhodnocení kritérií:
   a) Z04K1 - Spokojenost pracovníků
   Výsledky měření pro toto kritérium, které je založené na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů, je uvedené v následující tabulce:

Tabulka 5.8 – Z04K1 - Spokojenost pracovníků

	Body	Míra pociťovaného uspokojení
Z04K1	7	Míra uspokojení je dosti vysoká.

Zdroj: vlastní zpracování

b) Z04K2 - Využívání restrospektivy a zpětné vazby pro zlepšování v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace.
Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.9 – Z04K2 - Využívání restrospektivy a zpětné vazby pro zlepšování v oblastech: spolupráce, kvalitě produktů (služeb), procesů a organizace.

Kritérium	Otázka	Vyhod.
Z04K2O1	Používá organizace v softwarových projektech restrospektivu (jako u agilního rámce Scrum) a zpětnou vazbu za účelem podpory komunikace a dosažení zlepšování v oblastech: spolupráce, produktu a procesu?	Ne

Zdroj: vlastní zpracování

c) Z04K3 - Schopnost reprioritizace požadavků v rámci sprintu
Jedná se především o schopnost pružné reprioritizace požadavků nebo dodávek v rámci sprintu s ohledem na potřeby objednatele. Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.10 – Z04K3 - Schopnost reprioritizace požadavků v rámci sprintu

Kritérium	Otázka	Vyhod.
Z04K3O1	Je objednavatel schopný pružné reprioritizace požadavků nebo dodávek v rámci sprintu s ohledem na své potřeby?	Ne

Zdroj: vlastní zpracování

d) Z04K4 - Výměna poznatků mezi partnery
Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.11 – Z04K4 - Výměna poznatků mezi partnery

Kritérium	Otázka	Vyhod.
Z04K4O1	Probíhá mezi partnery výměna poznatků?	Ano

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení v této oblasti:
Na spokojenost pracovníků mají vliv ostatní oblasti, které je možné zlepšit pomocí ostatních navržených zásad. Velký vliv má to, že vývoj aplikace probíhá tradičním způsobem, který je řízeným plánem.
   - Přejít na agilní způsob vývoje softwaru.
   - Posílit vzájemnou komunikaci zavedením praktik: restrospektivy (jako u agilního rámce Scrum), zpětné vazby a vyhodnocování sprintů.
   - Zjistit návrhy na zlepšení v oblasti spolupráce a komunikace od samotných členů vývojového a provozního týmu.

5.1.5 Z05 - Vybrat vhodného partnera pro outsourcing

1. Zjištění, ověření a posouzení současného stavu
Proces výběrového řízení na určení dodavatele a vyjednání o smlouvě jsou řízeny centrálně útvarem Nákup. Tento útvar sjednocuje vstupy z banky a zajišťuje postupy dle předpisu *Určení dodavatele majetku a služeb*, že proces vyjednávání a smlouva jsou v souladu s předpisy

banky. Řízení na určení dodavatele probíhá separátně pro dodavatele softwaru a pro servis. Končí, když je podepsána smlouva nebo když se řízení na určení dodavatele zruší a je ukončena komunikace s účastníky.

Je nutné uvést před následujícím hodnocením, že hodnocená spolupráce a spokojenost s dodavatelem je postavena na softwarové projektu, který je řízen plánem (tradičně), běží už tři roky a u které se za tu dobu vystřídalo pět projektových manažerů.

2. Vyhodnocení kritérií:
    a) Z05K1 - Spolupráce dodavatele
    Výsledky měření pro toto kritérium, které je založené na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů, je uvedené v následující tabulce:

    Tabulka 5.12 - Z05K1 - Spolupráce dodavatele

	Body	Míra pociťovaného uspokojení
Z05K1	6	Spokojen mírně nadprůměrně.

Zdroj: vlastní zpracování

    b) Z05K2 - Spokojenost s dodavatelem
    Výsledky měření pro toto kritérium, které je založené na ordinální bodovací stupnici v rozsahu škály od 0 do 10 bodů, je uvedené v následující tabulce:

    Tabulka 5.13 - Z05K2 - Spokojenost s dodavatelem

	Body	Pociťované uspokojení
Z05K2	3	Spokojen v malé míře.

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení v této oblasti:
    - Zavést agilní způsob vývoje softwaru, například metodiku Scrum. V agilním světě je řízení na výběr dodavatele v zodpovědnosti role Scrum mastra. Multidisciplinární týmy jsou autonomní i z hlediska řízení dodavatelů.
    - Zaměřit se na roli Scrum mastra a předat mu pravomoc k řízení dodavatelů.
    - Zavést rituály jako zpětnou vazbu, retrospektivu a vyhodnocování sprintů do každodenní práce.
    - Soustředit se na zlepšování spolupráce na obou stranách, nejen na straně dodavatele v důsledku malé spokojenosti, tak i na straně objednavatele, aby se zlepšila kvalita spolupráce a komunikace.

5.1.6 Z06 - Zajistit znalost procesu softwarového vývoje v organizaci pro externího dodavatele

1. Zjištění, posouzení a ověření současného stavu:
V organizaci je popis IT procesů na vysoké úrovni. Jen sdílení těchto popisů je nedostatečné.
2. Vyhodnocení kritérií:
    a) Z06K1 - Znalost IT procesů
    Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.14 – Z06K1 - Znalost IT procesů

Kritérium	Otázka	Vyhod.
Z06K1O1	Má organizace popsány IT procesy?	Ano
Z06K1O2	Je organizace schopná prezentovat procesy vývoje softwaru a jeho dodávky externímu dodavateli?	Ano

b) Z06K2 - Proces vývoje softwaru
Vzhledem k tomu, že aplikace není vyvíjena agilně a že proces vývoje softwaru je řízen nějakou blíže nezjištěnou metodikou externího dodavatele, lze hodnotit toto kritérium, že není v souladu s agilním způsobem vývoje softwaru.

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Přejít na agilní způsob vývoje aplikace
   - Sdílet popis IT procesů

5.1.7 Z07 - Zajistit automatizaci rutinních procesů a zrychlit tok práce

1. Zjištění, posouzení a ověření současného stavu:
   Automatizace rutinních úloh je zavedena pro práci vývojářů ve vývojovém prosstředí prostřednictvím přístupu průběžné integrace (CI). Do integračního prostředí, které předchází předprodukčnímu prostředí, ale nasazují vývojáři ručně. Pro nasazení do předprodukčního prostředí je třeba navíc ručně zadat žádost do workflow nástroje Service Now a počkat na schválení a nasazení na základě výzvy od koordinátora releasů. Přestože existuje podpora pro automatizaci procesů v IT oblasti, není řízena, neexistuje plán.
2. Vyhodnocení kritérií:
   a) Z07K1 - Automatizace procesů
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

   Tabulka 5.15 – Z07K1 - Automatizace procesů

Kritérium	Otázka	Vyhod.
Z07K1O1	Jsou všechny podstatné procesy natrvalo automatizované?	Ne
Z07K1O2	Existuje podpora pro automatizaci procesů v IT oblasti?	Ano

Zdroj: vlastní zpracování

   V rámci daného softwarového projektu je míra automatizace procesů integrace, testování a nasazování softwaru podprůměrná. Na úrovni celé organizace je průměrná.

   b) Z07K2 – Průchodnost kódu do produkce
   Pro měření byly použity metriky průchodnosti kódu a porovnat je s vysoce výkonnými IT společnostmi (Puppet Labs, 2017):

Tabulka 5.16 – Z07K2 - Průchodnost kódu do produkce

Metriky průchodnosti	Organizace (objednavatel)	Vysoce výkonné IT společnosti (Puppet Labs, 2017)
Frekvence nasazování	Mezi jednou týdně a jednou za měsíc	Na požádání (vícenásobné nasazení za den)
Doba potřebná k realizaci změny	Mezi jednou týdně a jednou za měsíc	Méně než za hodinu
Doba potřebná do opravy chyby v produkci s ohledem na její dopad	Mezi několika dny (hotfix) a jednou za měsíc	Méně než za hodinu

Zdroj: vlastní zpracování

c) Z07K3 – Kvalita softwaru (chybovost)
Pro měření byly použity následující metriku chybovosti změn a porovnat ji s vysoce výkonnými IT společnostmi (Puppet Labs, 2017):

Tabulka 5.17 - Z07K3 – Kvalita softwaru (chybovost)

Metrika kvality softwaru	Organizace (objednavatel)	Vysoce výkonné IT společnosti (Puppet Labs, 2017)
Míra selhání změn	0-15%	0-15%

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Soustředit se na zajištění automatizaci rutinních a opakujících se úloh a procesů.
   - Zajistit podporu pro automatizaci, tzn. školení, čas, lidi a finanční prostředky.
   - Zavést přístup průběžného dodávání.

5.1.8 Z08 - Zajistit externím pracovníkům vývojových týmů přístup k interním prostředím a nástrojům

1. Zjištění, posouzení a ověření současného stavu:
   Externí vývojáři běžně používají přístup k interním prostředím a nástrojům organizace. Používají všechny tři způsoby přístupů: VPN, VDE a virtuální API. Nicméně zřízení přístupů, aby mohl člověk začít plnohodnotně pracovat, trvá v lepším případě několik dní (externí pracovník dostává pracovní laptop organizace). Není výjimkou, že to trvá i celý týden.

2. Vyhodnocení kritérií:
   a) Z08K1 - Sdílení interních prostředí a nástrojů s externím dodavatelem
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.18 – Z08K1 - Sdílení interních prostředí a nástrojů s externím dodavatelem

Kritérium	Otázka	Vyhod.
Z08K1O1	Sdílí organizace svá interní prostředí a nástroje s externími partnery?	Ano
Z08K1O2	Je možné nastavit sdílení interních prostředí a nástrojů během jediného pracovního dne?	Ne

Zdroj: vlastní zpracování

b) Z08K2 - Používané formy vzdáleného přístupu, jejich kvalitu a bezpečnost například formou interního IT auditu. Posoudit u jednotlivých forem, jak kvalitně může externista pracovat bez technických prodlev, výpadků techniky a sítí.
Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.19 – Z08K2 - Používané formy vzdáleného přístupu

Kritérium	Otázka	Vyhod.
Z08K2O1	Používají externí partneři organizace VPN pro přístup do interních prostředí?	Ano
Z08K2O2	Používají externí partneři organizace VDE pro přístup do interních prostředí?	Ano
Z08K2O3	Používají externí partneři organizace virtuální API pro přístup do interních prostředí?	Ano

Zdroj: vlastní zpracování

c) Z08K3 - Nedostupnost služeb
Pro měření byla zavedena metrika pro měření času, který externista stráví v důsledku nedostupných služeb.
Tuto metriku ale nebylo možné změřit stávajícími nástroji.

3. Navrhnout doporučení či změny za účelem zlepšení:
    - Zjednodušit a zrychlit nastavení sdílení interních prostředí a nástrojů tak, aby to bylo možné do jediného pracovního dne?

5.1.9 Z09 - Definovat metriky výkonnosti IT v organizaci a vyhodnocovat rychlost, kvalitu a efektivitu dodání softwaru od externího partnera

1. Zjištění, posouzení a ověření současného stavu:
Celkové měření výkonnosti IT v bance je dostupné jen IT manažerům. V této práci může být posouzena jen aplikace Generátor dokumentů, ke které jsou dostupná potřebná data a na jejímž vývoji se stále pracuje (integrace do systémů banky). Více informací o současné stavu je uvedeno ve vyhodnocení kritérií této zásady.
2. Vyhodnocení kritérií:
    a) Z09K1 - Výkonnost IT
    Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

Tabulka 5.20 – Z09K1 - Výkonnost IT

Kritérium	Otázka	Vyhod.
Z09K1O1	Měří se návratnost investice?	Ano
Z09K1O2	Měří se výkonnost IT podle metrik, které používají vysoce výkonné IT společnosti (Puppet)?	Ne
Z09K1O3	Měří se výkonnost IT jiný způsobem?	Ano
Z09K1O4	Používají se indikátory pro neustálé zlepšování nebo v rámci praktiky retrospektiva?	Ne

Zdroj: vlastní zpracování

b) Z09K2 - Průchodnost kódu do produkce
Pro měření byly použity metriky průchodnosti kódu a byly porovnány s vysoce výkonnými IT společnostmi:

Tabulka 5.21 – Z09K2 - Průchodnost kódu do produkce

Metriky průchodnosti	Organizace (objednavatel)	Vysoce výkonné IT společnosti (Puppet Labs, 2017)
Frekvence nasazování	Mezi jednou týdně a jednou za měsíc	Na požádání (vícenásobné nasazení za den)
Doba potřebná k realizaci změny	Mezi jednou týdně a jednou za měsíc	Méně než za hodinu
Počet chyb softwaru zjištěných v produkci	12	Méně než 8 chyb

Zdroj: vlastní zpracování

c) Z09K3 - Stabilita systémů
Pro měření byly použity metriky stability systémů a byly porovnány s vysoce výkonnými IT společnostmi:

Tabulka 5.22 – Z09K3 - Stabilita systémů

Metriky stability systémů	Organizace (objednavatel)	Vysoce výkonné IT společnosti (Puppet Labs, 2017)
Střední doba zotavení	Méně než za hodinu	Méně než za hodinu
Míra selhání změn	0-15%	0-15%

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Nastavit pro měření výkonnosti IT metriky a stability systémů, které používají vysoce výkonné IT společnosti (dle Puppet)
   - Používat se indikátory pro neustálé zlepšování nebo v rámci praktiky retrospektivy v rámci vývoje softwaru

5.1.10 Z10 - Řídit a spravovat dodávku softwaru od externího dodavatele pomocí přístupů CI, CDE a CD

1. Zjištění, posouzení a ověření současného stavu:
   Přístup průběžné integrace (CI) je implementován a plnohodnotně využíván. Nicméně ani jeden z přístupů průběžného dodávání (CDE) a plynulého nasazování (CD) není implementován.
2. Vyhodnocení kritérií:
   a) Z10K1 - Uplatnění přístupů průběžné integrace (CI), průběžného dodávání (CDE) a plynulého nasazování (CD)
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

   Tabulka 5.23 – Z10K1 - Uplatnění přístupů průběžné integrace (CI), průběžného dodávání (CDE) a plynulého nasazování (CD)

Kritérium	Otázka	Vyhod.
Z10K1O1	Má organizace zavedený přístup průběžné integrace (CI)?	Ano
Z10K1O2	Využívá organizace přístupu průběžného dodávání (CDE) pro dodávku softwaru?	Ne
Z10K1O3	Využívá organizace přístupu plynulého nasazování (CD) pro dodávku softwaru?	Ne

Zdroj: vlastní zpracování

   b) Z10K2 - Firemní kultura pro přístupy CI, CDE a CD
   Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

   Tabulka 5.24 - Z10K2 - Firemní kultura pro přístupy CI, CDE a CD

Kritérium	Otázka	Vyhod.
Z10K2O1	Má organizace firemní kulturu, která umožňuje pracovat s pomocí přístupu CI?	Ano
Z10K2O2	Má organizace firemní kulturu, která umožňuje pracovat v rámci přístupů CI a CDE?	Ne
Z10K2O3	Má organizace firemní kulturu, která umožňuje pracovat v rámci přístupů CI a CD?	Ne

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Změnit firemní kulturu organizace, aby umožnila zavést buď přístup CDE nebo CD podle typu aplikace
   - Implementovat buď přístup CDE nebo CD podle typu aplikace

5.1.11 Z11 - Zavést, spravovat a udržovat důvěryhodnost a bezpečnost softwaru na straně objednavatele

1. Zjištění, posouzení a ověření současného stavu:
   Spolupráce s externím dodavatelem trvá mnoho let. Za tu dobu už vytvořil nejméně jednu důležitou aplikaci, které zároveň zajišťuje servisní podporu (běží na infrastruktuře objednavatele). Vzhledem ke standardům organizace, co týká správy důvěryhodnosti,

a k řízení rizik, které je na vysoké úrovni, byla provedena následující klasifikace důvěryhodnosti dodavatele a softwaru.

2. Vyhodnocení kritérií:
   a) Z11K1 - Důvěryhodnost dodavatele
      Výsledky měření pro toto kritérium, které je založené na ordinální bodovací stupnici v rozsahu škály od 0 do 5 bodů, je uvedené v následující tabulce:

      Tabulka 5.25 – Z11K1 - Důvěryhodnost dodavatele

Kritérium	Body	Popis
Z11K1	4	Vysoká důvěryhodnost

Zdroj: vlastní zpracování

   b) Z11K2 - Důvěryhodnost softwaru
      Výsledky měření pro toto kritérium, které je založené na ordinální bodovací stupnici v rozsahu škály od 0 do 5 bodů, je uvedené v následující tabulce:

      Tabulka 5.26 – Z11K2 - Důvěryhodnost softwaru

Kritérium	Body	Popis
Z11K2	4	Vysoká důvěryhodnost

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   S ohledem na standardy organizace a vysokou úroveň řízení rizik nebyly navrženy žádné doporučení či změny za účelem zlepšení.

5.1.12 Z12 - Zavést controlling nákladů na straně objednavatele

1. Zjištění, posouzení a ověření současného stavu:
   Organizace má zavedený controlling nákladů v oblasti vývoje softwaru a jeho provozu, ale je nastaven na tradiční způsob vývoje software řízeného plánem.
2. Vyhodnocení kritérií:
   a) Z12K1 - Controlling nákladů v oblasti vývoje softwaru a jeho provozu
      Výsledky měření pro toto kritérium jsou uvedené v následující tabulce:

      Tabulka 5.27 – Z12K1 - Controlling nákladů v oblasti vývoje softwaru a jeho provozu

Kritérium	Otázka	Vyhod.
Z12K1O1	Má organizace zavedený controlling nákladů v oblasti vývoje softwaru a jeho provozu?	Ano
Z12K1O2	Má organizace zavedený controlling nákladů pro agilní způsob vývoje softwaru?	Ne

Zdroj: vlastní zpracování

3. Navrhnout doporučení či změny za účelem zlepšení:
   - Zavést controlling nákladů pro agilní způsob vývoje softwaru řízeného dodáním hodnoty pro byznys.

6 Závěr

Tato Diplomová práce se zaměřila na problematiku přístupu DevOps ve spojení s využitím outsourcingu vývoje a provozu softwaru.

Jak filozofie DevOps, tak i agilní způsob vývoje softwaru v kombinaci s využitím služeb externích softwarových společností znamenají změnu, a to naučit se dělat věci jinak, než tradičně. Vždy je nutné vědět, proč chceme změnu dělat, ať už se bude jednat o přechod na fungování podle DevOps principů, nebo na realizaci outsourcingu ICT procesů. Znát přínosy, ale uvědomit si i problémy a rizika, které mohou nastat. Pro outsourcing obecně platí, že rozhodnutí o něm jsou vždy zásadní („něco za něco“) a nevratná. Na druhou stranu nový software má zpravidla řešit problémy zákazníků, které jsou velmi rozmanité a vždy specifické pro svou doménu. V souvislosti s digitalizací různých oborů je zřejmé, že digitalizace donutí mnoho společností, aby ve stále větší míře využívaly ve svých nových aplikacích kód od externích třetích stran.

V úvodní části byl definován hlavní cíl a šest dílčích cílů. Hlavním cílem této diplomové práce bylo formulovat soubor zásad pro přístup DevOps v podmínkách outsourcingu vývoje softwaru, navrhnout vhodný model týmové organizační struktury a ověřit je v praxi.

Před splněním hlavního cíle práce bylo zapotřebí nejdříve splnit tři dílčí cíle práce, které tomuto hlavnímu cíli předchází. Prvním dílčím cílem DC1 teoretické části bylo charakterizovat přístup DevOps a s ním spojený agilní vývoj softwaru. Kromě charakteristiky DevOps a historie jeho vzniku jsem se zaměřil na to, jak je agilní vývoj softwaru integrován v DevOps. Pro charakteristiku agilního způsobu vývoje softwaru jsem zvolil nejpoužívanější agilní rámce, respektive procesní nástroje Scrum a Kanban, a Agilní manifest.

Druhým dílčím cílem DC2 byla podrobná analýza kultury, důvodů, klíčových principů, procesů od vývoje po provoz softwaru (Průběžná integrace, Plynulé nasazování atd.) a organizačních struktur, na nichž je založený přístup DevOps. Z lidského hlediska DevOps zdůrazňuje spolupráci, komunikaci a celkovou odpovědnost všech v celém životním cyklu softwaru. Z technického hlediska staví na automatizaci všech rutinních procesů, které se dělají pravidelně, a které zdržují od další práce. Současný stav technologií, nástrojů a schopností externích dodavatelů by měl přimět vedoucí pracovníky IT k tomu, aby přijali přístup DevOps pro vývoj softwaru. Pro jeho úspěch je nutný myšlenkový posun u lidí skrze změnu pracovní kultury, kterou je třeba provádět ve vlnách a to tak, že se změní pravidla, na niž stojí kultura organizace. DevOps spolu s dalšími integrovanými přístupy Průběžné integrace (CI), Průběžného dodávání (CDE), Plynulého nasazování (CD) podporují agilní způsoby vývoje softwaru a zároveň řeší komplexně problematiku IT provozu a údržby softwaru. Tato kombinace přístupů, postupů a metodik se ujala celosvětově a neustále expanduje.

Třetím dílčím cílem DC3 byla stručná charakteristika částečného outsourcingu IS/ICT, a to jednoho z ICT procesů, vývoje softwaru, jako služby poskytované třetí stranou. Pro splnění tohoto cíle jsem se dále zabýval postupem pro výběr vhodného dodavatele, smluvními vztahy a identifikací rizik spojených s tímto typem outsourcingu. Kromě toho jsem uvedl i kritické faktory externí dodávky. Tímto způsobem lze získat skvělé motivované profesionály, kteří jsou odborně a technicky zdatní a mají zkušenosti. Více než kdy předtím existují schopné týmy programátorů po celém světě, které může organizace využít pro vývoj úspěšného software.

Dílčím cílem DC4 bylo formulovat zásady pro fungování DevOps a outsourcingu vývoje softwaru. Zaměřil jsem se na řešení, přístupy, které mají přispět k efektivnímu fungování partnerství, kvalitní spolupráci a k neustálému zlepšování v oblastech: spolupráce, kvality produktů (služeb), procesů a organizace, které povede k vývoji a provozu úspěšného softwaru. Je třeba si neustále uvědomovat tradiční bariéry mezi vývojem a provozem softwaru, ale i nově vznikající bariéry například vůči externím dodavatelům a překonávat je. V zásadě mohu konstatovat, že jsem nenašel žádný důvod, proč by DevOps nemohl pracovat s externími dodavateli.

V současné době je vývoj softwaru týmová hra a tomu bylo nutné přizpůsobit řízení organizace a týmovou organizaci. Dílčím cílem DC5 byl proto návrh jednoho nebo více vhodných modelů týmové organizační struktury pro spojení DevOps a outsourcingu vývoje softwaru. Dále pak stanovení metrik a definice přístupu k důvěře a bezpečnosti mezi organizací a třetí stranou. Tento cíl byl splněn návrhem modelů. Je třeba uvést, že pro úplnost jsem přidal další dva modely, které organizace využívají a které jsou vhodné pro spojení obou konceptů.

Pokud chce být organizace vysoce výkonná v oblasti IS/ICT, je to nemyslitelné bez radikální změny organizační struktury a řízení organizace. Cílem DevOps jsou multidisciplinární týmy zodpovědné za konkrétní služby nebo aplikace a týmy platforem, které jim budou poskytovat své produkty formou samoobslužných služeb. Na základě mého výzkumu organizačních struktur mohu konstatovat, že nelze vyloučit, že budou vyvinuty další nové modely týmových organizačních struktur, které budou reflektovat nové trendy v oblasti vývoje softwaru a provozu IT.

Závěrem práce bylo také nutno splnit šestý dílčí cíl práce DC6, a to ověřit navržené zásady a postupy DevOps v podmínkách outsourcingu vývoje softwaru v praxi. Tento cíl byl splněn demonstrací aplikace zásad v projektu vývoje aplikace pro tvorbu dokumentace.

Vzhledem k tomu, že DevOps dalo vzniknout hnutí IT profesionálů, bude se tento koncept dále vyvíjet a s velkou pravděpodobností se bude zabývat i tím, jak řešit problémy při spolupráci s externími dodavateli. Proto lze očekávat, že zásady pro práci s externími dodavateli najdou své uplatnění. Filozofie DevOps, agilní přístupy a Lean byly nově začleněny do ITIL verze 4, která se dosud o oblasti vývoje softwaru nezmiňovala. Dalším možným tématem rozšíření této práce tedy může být, jak řízení IT služeb spolupracuje s přístupy DevOps, agilními rámci a Lean a jak to celé bude zapadat do digitální transformace.

Přijetí kultury DevOps samo o sobě nebo navíc ještě posílené o spolupráci s vysoce výkonnými externími IT firmami či poskytovateli služeb se může stát silnou stránkou organizace, kterou poté transformuje do vytvoření konkurenční výhody pro celý podnik.

Na DevOps mě nejvíce zaujalo to, že uvádí v život Fail Fast princip. Tento princip znamená, že multidisciplinární tým zkouší, chyb se nebojí, naopak s nimi počítá a učí se z nich. Zároveň opouští svou komfortní zónu a provádí experimenty, které začínají hypotézou, kterou pak pomocí experimentu ověří. V praxi to znamená, že když vidí první rizikovou věc, je to první, do čeho se pustí pomocí experimentu. Není tak nikdy hotovo a vždy je nutné usilovat o další zlepšování.

Vzhledem k výše uvedenému konstatuji, že se mi podařilo naplnit zadaný cíl práce a v rámci rozšíření této práce je možné soubor zásad pro zavedení DevOps v podmínkách outsourcingu vývoje softwaru dále postupně zdokonalovat. Například tak, že se hodnocení splnění zásad rozšíří

o další kritéria, nebo že se zpřístupní komunitě odborníků v IT oblasti, jež by pak měla hrát důležitou roli, formou spolupráce, při jeho dalším vylepšování a rozšiřování.

Terminologický slovník

Termín	Zkratka	Význam (zdroj)
Application Services Providing	ASP	Poskytování aplikačních služeb. V oblasti podnikových informačních systémů jedna z forem outsourcingu. (Voříšek, a další, 2004)
Application Programming Interface	API	Aplikační programovací rozhraní; API se říká rozhraní, které programátor využívá pro komunikaci se softwarem. Jedná se o balík knihoven, funkcí či nějakých procedur, které může programátor využívat a ovládat či komunikovat se softwarem. (IT-slovnik.cz)
Bodyshopping		Najmutí odborníků v oblasti IT. (autor)
Continuous Integration	CI	Průběžná integrace je praktika ve vývoji softwaru, kdy členové týmu často integrují svou práci, obvykle každý člověk integruje alespoň denně - což vede k více integracím za den. Každá integrace je ověřena automatizovaným buildem (včetně testování), ve kterém se co nejrychleji detekují integrační chyby. (Fowler, 2006)
Continuous Delivery	CDE / CD	Průběžné dodávání; Implementace průběžného dodávání znamená zajistit, že software je vždy připraven do produkce po celý jeho životní cyklus – že jakýkoli build by mohl být uvolněn uživatelům stisknutím tlačítka s pomocí plně automatizovaného procesu během několika sekund nebo minut. (Humble a Farley, 2010)
Continuous Deployment	CD	Plynulé nasazování spočívá v průběžném automatizovaném nasazování změn přímo do produkčního prostředí. Charakteristické pro plynulé nasazování jsou časté releasy, které procházejí sérií automatizovaných testů. (autor)
DevOps	DevOps	DevOps je akronym anglického slovního spojení Development a Operations. Označuje se jím koncept, který usiluje o spojení či určitý stupeň integrace dvou dříve zcela oddělených IT oblastí – vývoje (Dev) a provozu (Ops). (autor)
Gartner		Gartner, Inc. je americká společnost, která se zabývá se výzkumem a poradenstvím v oblasti IS/ICT technologií. Do roku 2001 vystupovala pod názvem Gartner Group, Inc. (autor)
Infrastructure-as-a-Service	IaaS	Infrastruktura jako služba; IaaS poskytuje výpočetní techniku pro podporu podnikových operací na základě outsourcingu. IaaS poskytuje nejčastěji hardware, úložiště, servery a prostory datového centra nebo síťové komponenty. Může ale také zahrnovat software.(IT-slovnik.cz)
Kanban		Agilní rámec, resp. procesní nástroj, kter založený na vizualizaci pracovního postupu, na základě rozpadávání pracovních úkolů na menší části a jejich evidenci na pracovní tabuli (viz Kanban board). (Kniberg a Skarin, 2010, s. 4-7)
Multidimensional Management and Development of Information Systems	MMDIS	MMDIS je metodika přístupu k vývoji informačních systémů. Tato metodika je od roku 1990 vyvíjena na Fakultě informatiky a statistiky, VŠE v Praze. (autor)

Outsourcing		Dlouhodobý smluvní vztah s externím subjektem, který organizaci poskytuje služby v jedné nebo více funkčních oblastech. (autor)
Platform-as-a-Service	PaaS	Platforma jako služba; PaaS popisuje výpočetní platformu, která se pronajímá nebo je dodávána jako integrované řešení či služba prostřednictvím připojení k internetu. (IT-slovnik.cz)
Prediktor výkonnosti IT		Prediktory výkonnosti IT jsou nástroje, procesy, pravidla nebo kultura, které předpovídají, že se IT v budoucnu stane více výkonným, nebo to bude jejich důsledkem.
Request for information	RFI	Žádost o dodatečné informace.
Request for	RFP	Žádost o nabídku.
Return On Investments	ROI	ROI vyjadřuje návratnost investice. Definuje poměr mezi výnosem a investovanými prostředky. (autor)
SCRUM		Metoda agilního vývoje, která týmům dovoluje v krátkých iteracích vyvíjet produkty s dobře odhadnutelnou cenou a podle zákaznických požadavků. (autor)
Software-as-a-Service	SaaS	Software jako služba; SaaS je služba, která se týká poskytování softwaru, který je využíván přes webové rozhraní. Jelikož se při této službě software hostuje na dálku, uživatelé nemusí investovat do dalšího hardwaru. SaaS odstraňuje u firem nutnost instalovat, nastavovat nebo aktualizace daný software. (IT-slovnik.cz)

Použité informační zdroje

BESTPRACTICE.CZ. *Vlastnosti prostředí DevOps* [online]. OMNICOM s.r.o., 2017 [cit. 2019-03-29]. Dostupné z: https://www.bestpractice.cz/cs/Best-practice/DevOps/Vlastnosti-prostredi-DevOps.alej

BRUCKNER, Tomáš, VOŘÍŠEK, Jiří, BUCHALCEVOVÁ, Alena, STANOVSKÁ, Iva, CHLAPEK, Dušan a ŘEPA, Václav. *Tvorba informačních systémů: principy, metodiky, architektury*. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-2474153-6.

DEVOPS AGILE SKILLS ASSOCIATION. *DASA DevOps Fundamentals*. R1.3.0. Praha: GOPAS. 2018. Course eBook.

DUVALL, Paul M, MATYAS, Steve a GLOVER, Andrew. *Continuous integration: improving software quality and reducing risk*. Upper Saddle River, NJ: Addison-Wesley, 2007. ISBN 978-0-321-33638-5.

GARTNER. *Creating a Sustainable Strategy for Sourcing at Midsize Enterprises* [online]. Gartner, 2018. [cit. 2019-03-29]. Dostupné z: https://www.gartner.com/document/3877164?ref=solrAll&refval=209673357&qid=fe1ccda43e07e65cc39364734fc74c21

GARTNER. *Managing Digital Trust in the Software Development Life Cycle* [online]. Gartner, 2017. [cit. 2019-03-29]. Dostupné z: https://www.gartner.com/doc/3730417/managing-digital-trust-software-development

GARTNER. *Gartner on Outsourcing* [online]. Gartner, 2005. [cit. 2019-03-29]. Dostupné z: https://fenix.tecnico.ulisboa.pt/downloadFile/3779576866179/Gartner-Outsourcing.pdf

FARENHORST, Rik a LOADER, Niels. *Embracing Digital Disruption by Adopting DevOps Practices* [online]. DevOps Agile Skills Association (DASA), 2016. [cit. 2019-03-29]. Whitepaper. Dostupné z: http://www2.itpreneurs.com/devopswhitepaper

FOWLER, Martin. Continuous Integration. In: *MartinFowler.com* [online]. 2006-05-01 [cit. 2019-03-29]. Dostupné z: https://martinfowler.com/articles/continuousIntegration.html

FOWLER, Martin. Continuous Delivery. In: *MartinFowler.com* [online]. 2013-05-30 [cit. 2019-03-29]. Dostupné z: https://martinfowler.com/bliki/ContinuousDelivery.html

HUMBLE, Jez a David FARLEY. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Upper Saddle River, NJ: Addison-Wesley, 2010. ISBN 9780321601919.

HÜTTERMANN, Michael. *DevOps for developers. Integrate Development and Operations, The Agile Way*. New York: Distributed to the book trade worldwide by Springer Science+Business Media New York, 2012. Expert's voice in Web development. ISBN 978-1-4302-4569-8.

ISACA. *DEVOPS OVERVIEW* [online]. Rolling Meadows, USA: ISACA, 2015. [cit. 2019-03-29]. An ISACA DevOps Series White Paper. Dostupné z: https://informationsecurity.report/Resources/Whitepapers/8a0480cd-71a2-4cc0-98bd-f1ce2e8f03e4_DevOps_whp_Eng_0115.pdf

IT-slovnik.cz. IT Slovník.cz [online]. [cit. 2019-03-29]. Dostupné z: https://it-slovnik.cz/

JANSA, Lukáš, OTEVŘEL, Petr a ŠTEVKO, Martin. *Softwarové právo. 3. aktualizované a rozšířené vydání*. Brno: Computer Press, 2018. ISBN 9788025149140.

KIM, Gene, DEBOIS, Patrick, WILLIS, John, HUMBLE, Jez a ALLSPAW, John. *The DevOps handbook: how to create world-class agility, reliability, & security in technology organizations*. Portland, OR: IT Revolution Press, 2016. ISBN 978-1942788003.

KIM, Gene. *The top 11 things you need to know about DevOps* [online]. IT Revolution Press, 2013. [cit. 2019-03-29]. Dostupné z: https://www.thinkhdi.com/~/media/HDICorp/Files/White-Papers/whtppr-1112-devops-kim.pdf

KNIBERG, Henrik a SKARIN, Mattias. *Kanban and Scrum making the most of both*. InfoQ.com, Enterprise Software Development series. C4 Media, 2010. ISBN 978-0-557-13832-6.

LEFFINGWELL, Dean. *Scaling software agility: best practices for large enterprises*. Upper Saddle River: Addison-Wesley, 2007. The Agile software development series. ISBN 0-321-45819-2.

MACHÁČKOVÁ, Eva a MACHÁČEK, Zdeněk. KANBAN pro efektivní řízení práce v rámci týmu. *IT SYSTEMS* [online]. 2017, roč. 17, č. 1-2 [cit. 2019-03-29]. Dostupný z: https://www.systemonline.cz/rizeni-projektu/kanban-pro-efektivni-rizeni-prace-v-ramci-tymu.htm

PAUL, Fredric. Gene Kim Explains 'Why DevOps Matters'. In: Blog.NewRelic.com [online]. 2015-06-24 [cit. 2019-03-29]. Dostupné z: https://blog.newrelic.com/technology/gene-kim-why-devops-matters/

PAUL, Fredric. Performance Monitoring for Digital Businesses. In: Blog.NewRelic.com [online]. 2014-05-16 [cit. 2019-03-29]. Dostupné z: https://blog.newrelic.com/2014/05/16/devops-name/

PUPPET LABS. *2014 State of DevOps Report* [online]. Puppet Labs, 2014. [cit. 2019-03-29]. Dostupné z: https://puppet.com/resources/whitepaper/2014-state-devops-report

PUPPET LABS. *2016 State of DevOps Report* [online]. Puppet Labs, 2016. [cit. 2019-03-29]. Dostupné z: https://puppet.com/resources/whitepaper/2016-state-devops-report

PUPPET LABS. *2017 State of DevOps Report* [online]. Puppet Labs, 2017. [cit. 2019-03-29]. Dostupné z: https://puppet.com/resources/whitepaper/2017-state-of-devops-report

PUPPET LABS. *2018 State of DevOps Report* [online]. Puppet Labs, 2018. [cit. 2019-04-10]. Dostupné z: https://puppet.com/resources/whitepaper/state-of-devops-report

SLATINSKÝ, Karel. *Integrace agilních a DevOps přístupů do ITIL*. Brno, 2016. Diplomová práce. Masarykova univerzita v Brně, Fakulta informatiky. Vedoucí práce: Ing. RNDr. Barbora Bühnová, Ph.D. Dostupné také z: https://is.muni.cz/th/pvstx/Karel_Slatinsky-Diplomova_prace.pdf.

SCHWABER, Ken and SUTHERLAND, Jeff. *The Scrum Guide™: The Definitive Guide to Scrum: The Rules of the Game* [online]. USA: 2017. [cit. 2019-03-29]. Dostupné z: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

THE AGILE ALLIANCE. Manifest Agilního vývoje software. In: Agilemanifesto.org [online]. Překl. Zuzana Šochová a kol. 2001 [cit. 2019-03-29]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

VOŘÍŠEK, Jiří, PAVELKA, Jan, a VÍT, Miroslav. *Aplikační služby IS/ICT formou ASP: proč a jak pronajímat informatické služby*. Praha: Grada, 2004. Management v informační společnosti. ISBN 80-247-0620-2.

Seznam obrázků a tabulek

Seznam tabulek

Seznam obrázků