Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	LS2017
Autoři	Zbyněk Zelinka, xzelz06 Miroslav Horňák, xhorm90
Téma	Use DevOps to Drive Your Agile ALM
Datum odevzdání	14.5.2017

Abstrakt

V této seminární práci je nejprve nastíněna problematika řízení životního cyklu aplikace (ALM), dále se snaží poukázat na rozdíly mezi klasickým ALM a agilní přístupem k ALM a uvést několik z nejpoužívanějších nástrojů. V další části definovat pojem DevOps, jaké je jeho využití, jaké problémy řeší, jaké jsou dostupné nástroje a jak fungují. Nakonec bude poukázáno, jak využít praktik a postupů obsažených v DevOps pro úspěšné řízení agilního ALM a budou stanoveny 4 klíčové faktory úspěchu jejich implementace. V závěru dojde ke shrnutí zmíněných témat a zhodnocení cílů práce.

Klíčová slova
ALM, Agile ALM, DevOps, Chef, Lifecycle, Software, Development

Obsah

1. Úvod

Vývoj podnikových aplikaci je velmi komplexní úkol zahrnující sledování a řízení velkých skupin lidi, dodržování milníků a komplexních závislostí, které je obtížné sledovat na každodenním bázi. Pokud toho je však docíleno, pozitivní výsledky mohou být enormní, a proto je toto téma tak důležité a je třeba se jím zabývat. Tato práce se pokusí definovat pojmy jako SDLC, ALM nebo DevOps, které mohou s tímto obtížným úkolem pomoci, poukázat, jak spolu souvisí a podat návod, jak s jejich pomocí dosáhnout kýžených výsledků. Dále práce uvádí příklady konkrétních nástrojů, jenž lze využít a na konec uvádí několik klíčových oblastí, na které se je třeba zaměřit při implementaci těchto metodik v podniku.

2. Application Lifecycle Management

Pod zkratkou ALM se skrývá pojem Application LIfecycle Management, což by se dalo přeložit jako řízení životního cyklu aplikace a jedná se tedy o odnož Řízení životního cyklu produktu (PLM) zabývající se softwarem. (Yllemo, 2016)

Životní cyklus aplikace začíná dříve, než by se mohlo na první pohled zdát. Jedna z definic popisuje životní cyklus aplikace jako časový úsek, kdy společnost vydává finanční prostředky na tuto aplikaci, tzn. celou dobu kdy aplikace společnost něco stojí. Z pohledu ALM začíná životní cyklus aplikace totiž již v moment, kdy dojde k vznesení požadavku na vytvoření aplikace, případně se objeví nápad na realizaci nového softwaru. Aplikace poté prochází různými fázemi svého životního cyklu, až je nakonec vyřazena z provozu, čímž její životní cyklus končí. Zde je dobře vidět rozdíl oproti přístupu Software Development Lifecycle (SDLC), který se zaměřuje pouze na část životního cyklu od počátku po vydání aplikace a jeden ALM cyklus tak může obsahovat (a zpravidla také obsahuje) hned několik SDLC cyklů. ALM by se dalo charakterizovat jako nástroj, který pomáhá přetvářet business požadavky společnosti na hotový software. Má velice široký scope, jelikož se snaží identifikovat všechny, nebo co možná nejvíc zúčastněných osob a stran a pomoci s komunikací a spolupráci mezi nimi. (Yllemo, 2016; Aiello, a další, 2014)

ALM se skládá ze tří hlavních oblastí, které se snaží co nejlépe integrovat a napomáhat jejich vzájemné spolupráci. Tyto oblasti jsou následující:

2.1. Governance

Jako jediná oblast se napíná napříč celým životním cyklem aplikace. Pod governance se skrývá celá řada činností spojených s řízením provozu nebo vývoje, rozhodováním apod. Jednotlivé činnosti jsou také rozděleny podle aktuální fáze životního cyklu, např. v počáteční fázi se jedná takřka výhradně o sestavení business case, během vývoje je to Project portfolio management, a nakonec během provozu aplikace se činnosti řadí především do Apllication portfolio managementu. (Yllemo, 2016)

2.2. Development

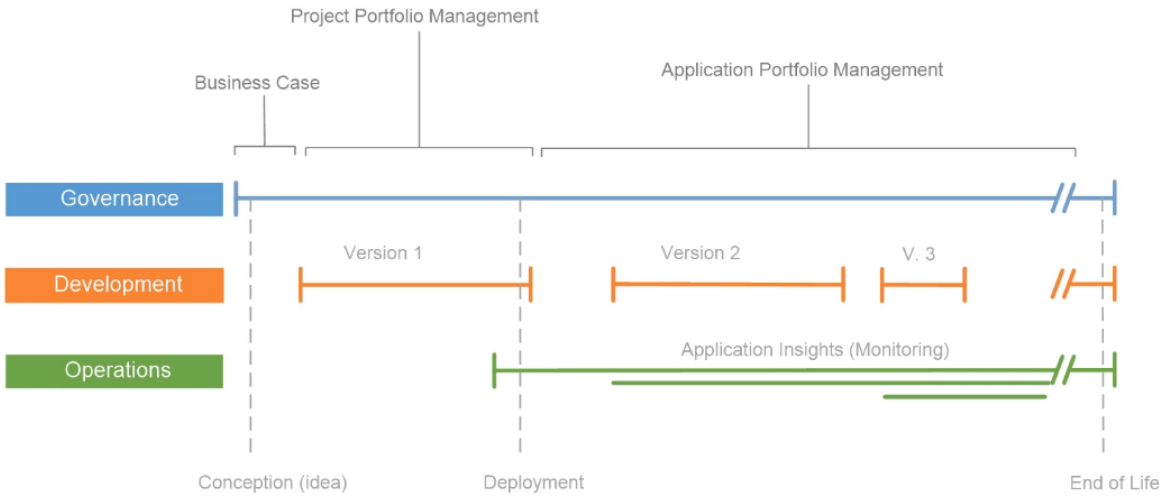
Development, nebo česky vývoj je, jak už název napovídá, zodpovědný za vytvoření, otestování a uvolnění celé aplikace. Je tedy oblastí dominantní především v raných fázích životního cyklu aplikace, nicméně její význam nekončí po vydání aplikace. Vývojová linka totiž tímto milníkem nekončí, neboť aplikace zpravidla potřebují průběžně aktualizovat, což opět spadá pod záštitu vývojářů. Zde je dobře vidět rozdíl mezi ALM a dříve zmíněným SDLC, který pokrývá pouze vývoj, a navíc pouze v rámci jednoho vývojového kroku, tzn. pro každý následující update existuje separátní SDLC. (Yllemo, 2016)

2.3. Operations

Operations neboli provoz je oblastí zodpovědnou za bezproblémový a bezporuchový běh aplikace a zajištění maximálního komfortu pro uživatele spravované aplikace. Oblast je u většiny aplikací dominantní během většiny jejího životní cyklu od vydání až po vyřazení z provozu a je tedy většinou i největší položkou co se týče výsledné ceny aplikace. Problematické může být například provozování více verzí aplikace současně. (Yllemo, 2016)

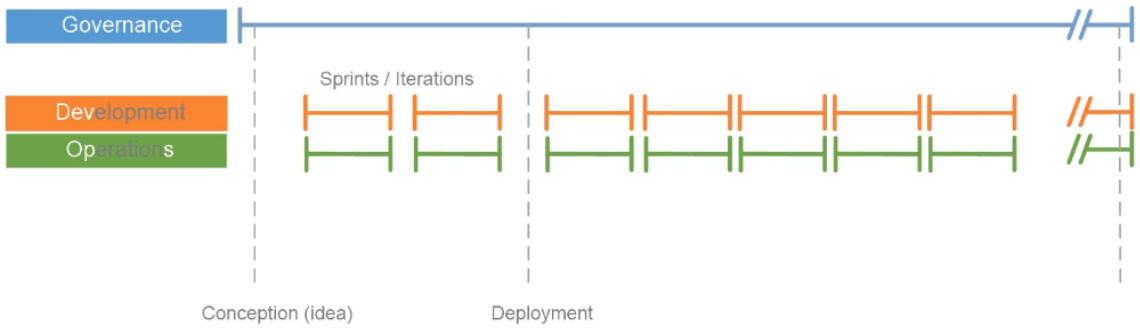
3. ALM a Agile ALM

Aplikováním agilního přístupu na klasický ALM vznikla metodika Agile ALM, která předčila svého předchůdce ve výkonnosti i oblíbenosti mezi podniky. Jak se ukázalo aplikování agilních metodik vede v této oblasti k zvýšení kvality výsledného produktu, ale i celkové spokojenosti na straně dodavatele. V této kapitole budou nyní demonstrovány rozdíly mezi ALM a Agile ALM a bude poukázáno na výhody, které agilní ALM nabízí. (Yllemo, 2016)



Obrázek 1 Klasické ALM (Zdroj: Almbok, 2017)

První obrázek ilustruje pohled na klasické ALM s jeho třemi typickými oblastmi. Obrázek dobře ilustruje důležitost vývoje i po vydání aplikace. V oblasti provozu si lze všimnout, že v příkladu je spravováno hned několik verzí aplikace, každá reprezentována jednou z rovnoběžných čar, přičemž každá z nich začíná právě na konci jednoho z cyklů vývoje. (Yllemo, 2016)



Obrázek 2 Agilní ALM (Zdroj: Almbok, 2017)

Druhý obrázek naopak představuje implementaci agilního ALM. Oproti předchozímu obrázku je největším rozdílem spojení vývoje a provozu do jedné společné linie a provádění krátkých iterací (tzv. sprintů) společně. Cílem tohoto přístupu je dodat co nejrychleji nějakou

hodnotu zákazníkovi systému a na projekt se dá nahlížet i jako na nekončící lineární funkci. Tato spolupráce mezi odděleními má pozitivní dopad na obě skupiny zejména díky lepší výměně informací a snadnější komunikaci. (Yllemo, 2016)

4. ALM nástroje

ALM nástroje jsou často dodávány jako sady nástrojů, které lze používat i individuálně, nicméně teprve jako celek můžou využít celý svůj potenciál .Na trhu se nachází množství různých takovýchto sad a vybrat to správné řešení, aby se z investice nestal "shelfware", který nikdo nepoužívá, ale aby nástroje společnosti skutečně přinesly požadovaný užitek, může někdy být velice obtížné, zejména vezme-li se v potaz fakt, že ve většině společností už existují aplikace, které bude třeba integrovat, nebo nahradit.
Mezi nejsilnější hráče na trhu patří:

4.1.1. HP Application Lifecycle Management (HP ALM) od HP

HP ALM je silný nástroj zaměřený především na absolutní průhlednost a pořádek v procesech životního cyklu aplikace. Výhodou oproti konkurenci je relativně snadná integrace s produkty od jiných dodavatelů. Nedávno HP také umožnilo využívat HP ALM vzdáleně jako SaaS (Software as a service). (Owen, 2017)

4.1.2. Visual Studio Team Foundation server (TFS) od Microsoftu

ALM sada od Microsoftu je založena na jednom z nejoblíbenějších vývojářském prostředí – Visual studiu. To znamená, že větší společnosti již často používají některý z nástrojů této sady. Mezi největší předností patří zejména pokročilé nástroje pro řízení verzí a konfigurací. (Owen, 2017)

4.1.3. Rational solution for Collaborative Lifecycle Management (CLM) od IBM

CLM je standardem na trhu ALM zejména díky integraci s ostatními podnikovými systémy od IBM. Podporuje agilní i klasický vodopádový model vývoje a ačkoli se jedná o starší produkt s řadou dnes už obstarožních funkcí, IBM udržuje produkt dostatečně aktuální, aby obstál i současným požadavkům o čemž vypovídá jeho stálá popularita. (Owen, 2017)

Mezi další nástroje patří např. GitLab (GitLab), FusionForge (SourceForge) nebo Endevor (CA Technologies). (Owen, 2017)

5. DevOps

Za DevOps lze považovat kombinaci společenské filosofie, přístupů a nástrojů, které zvyšují schopnost organizací vytvořit aplikace a služby co nejrychleji. S DevOps dokáže firma zlepšovat produkt v mnohem kratším čase, než by bylo možné s tradičním přístupem vývoje omezeným procesy pro řízení infrastruktury. Díky velmi rychlému vývoji nových funkcí a verzí je možné zákazníkům doručit nový produkt rychleji, čímž dokáže společnost výrazně zvýšit svoji konkurenceschopnost na trhu. (Amazon, 2017)

Jak už z názvu vypovídá, základním kamenem pro DevOps je spojení vývoje (development) a provozu (operations) do jednoho společně fungujícího ekosystému. Každé oddělení má své vlastní problémy a požadavky, spojují je však stejné věci – tedy snaha vytvořit kvalitní aplikaci a s tím spojena potřeba vytvářet a spravovat nové verze softwaru. (Gene, 2013)

5.1. Problém

Z pohledu vývoje je v tradičním přístupu hned několik úskalí. Prvním z nich je čas do vydání nové verze softwaru, který někdy může být až několik týdnů. Vývojář tedy opraví nahlášenou chybu a na vydání této opravy čeká až do oficiálního termínu. Mezitím však pracuje na nových funkcích a ztrácí přehled, jaké změny provedl na začátku měsíce. Pokud se pak při vydání zjistí, že udělal někde chybu, musí se ke staré práci vracet, což je nejen časově náročné, ale zároveň i značně demotivující. Dalším problémem je pak rozdíl mezi vývojovým a produkčním prostředí. Může se tak jednoduše stát, že vývojář upravil chybu nad rozhraním, které už někdo jiný změnil a na poslední chvíli musí svoji práci upravovat, aby stihl uzávěrku nové verze. (Rackspace, 2013)

Na druhé straně je pak provoz neboli administrace systému, která má za úkol udržovat servery v neustálém chodu a zajistit kvalitu nového softwaru. Je právě jejich zodpovědností, aby nasazení nové verze proběhlo v pořádku. S rostoucím množstvím serverů, které mají často rozdílnou konfiguraci, se stává práce správců čím dál tím složitější. Dříve dostačující nástroje již na tak velké množství serverů nestačí a pomocí starých nástrojů musí i někdy manuálně nasazovat novou verzi na každý server zvlášť. Udržet tak spolehlivost systému je téměř nemožný úkol. Nasazení nové verze je pro ně tak složité, že se uchýlili ke kompromisu, že bude tento proces provádět pouze jednou za měsíc. (Rackspace 2013)
V den vydání nové verze musí být provozem ověřeno, že je vše připraveno k nasazení. Pokud najdou nějaký problém při buildu této verze, tak musí kontaktovat vývoj, aby tento problém urychleně opravil. Na základě tohoto požadavku se pracovník vývoje uvolní k opravě této chyby a pošle novou verzi zpět na provoz. Takto se to opakuje, dokud není vše v pořádku a pracovníci si tak často připadají, jako by jim vývoj úmyslně házel klacky pod nohy. Na druhé straně vývoj by rád pracoval na nových funkcích další verze, čímž se práce obou táborů značně komplikuje. S tím vším samozřejmě rostou i náklady a čas strávený nad každým nasazením nové verze. (Rackspace, 2013)

5.2. Řešení problému pomocí DevOps

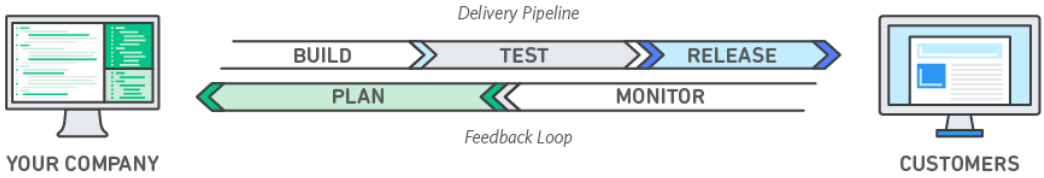
Práce obou táborů směřuje ke stejnému cíli, tedy ke spokojenosti zákazníků, ale také i k vlastní spokojenost. DevOps se snaží spojením těchto dvou táborů dosáhnout vzájemné pomoci a společného myšlení, díky kterému tento problém mohou společně vyřešit. Společnou prací a myšlením mohou vytvořit nové aktualizace rychleji a společně sdílet odpovědnost za jejich funkčnost. V tomto modelu se v některých případech tyto týmy spojují do jednoho týmu, ve kterém společně pracují napříč celým životním cyklem aplikace. K tomu se váží i týmy kvality a bezpečnosti, které jsou taktéž směřovány ke společné spolupráci s vývojem a provozem. (Rackspace, 2013)

Klíčem úspěchu DevOps je automatizace infrastruktury, které vede ke zvýšení produktivity a k potenciální vyšší kvalitě softwaru. Automatizací se rozumí nejen automatizace na straně vývoje, jako je testování kódu, ale i celkově automatizování workflow, procesů a v podstatě všeho, co se opakuje. Toho je možné dosáhnout správným pochopením metodik, společným úsilím, které je dobré podpořit specializovaným softwarem. Zdaleka nejdůležitějším faktorem je ochota týmů spolupracovat a přizpůsobit se aspektům DevOps. (Rackspace, 2013)

Další důležitou částí DevOps je přístup k vývoji, kdy je oproti tradičnímu přístupu snaha vytvářet nové verze co nejčastěji a v malých dávkách. Díky tomu se vývojáři mohou soustředit na problémy v reálném čase a měřit dopad jejich práce na celkový výkon softwaru. Právě měření výkonu je jednou z dalších domén DevOps, kdy je vývoj v malých balíčcích mnohem jednodušší sledovat a následně upravit daný balíček. (Amazon, 2017)

V neposlední řadě je velmi výhodné upravit architekturu softwaru tak, aby se skládal z co nejmenších služeb. Aplikace je tedy roztříštěna do několika komponent (služeb), které slouží jednomu specifickému účelu a operují téměř samostatně. Tím je organizaci umožněno pracovat na jednotlivých službách zvlášť a snižuje se tak potřeba spojování jednotlivých funkcí a verzí. V praxi tak může být každý agilní tým ve společnosti zodpovědný za jednu službu a celkově je možné pracovat mnohem efektivněji. (Amazon, 2017)

Menší balíčky aktualizací a rozdělení aplikace do služeb znamená mnohem větší množství nasazení aplikací, což znamená poměrně velký tlak na provoz. To ovšem DevOps řeší průběžnou integrací a nasazením softwaru a dovoluje tak společnosti vytvářet rychle rostoucí a bezpečný produkt. (Amazon 2017) Ve spojení s automatizací a upravenou architekturou dokáže společnost udržovat software ve stavu, kde je jej možné velmi dobře udržovat a provádět na něm změny. Společně tak tyto postupy dopomůžou společnosti k doručení aktualizace ke koncovému zákazníkovi rychleji s ještě vyšší spolehlivostí. Kromě spokojenějšího zákazníka roste i konkurence schopnost společnosti na trhu, která právě díky rychlejšímu vývoji a nasazení může přijít na trh s novou verzí produktu rychleji než konkurence. (Harvey, 2017)



Obrázek 3 Workflow DevOps (Zdroj: Amazon, 2017)

5.2.1. Výhody DevOps

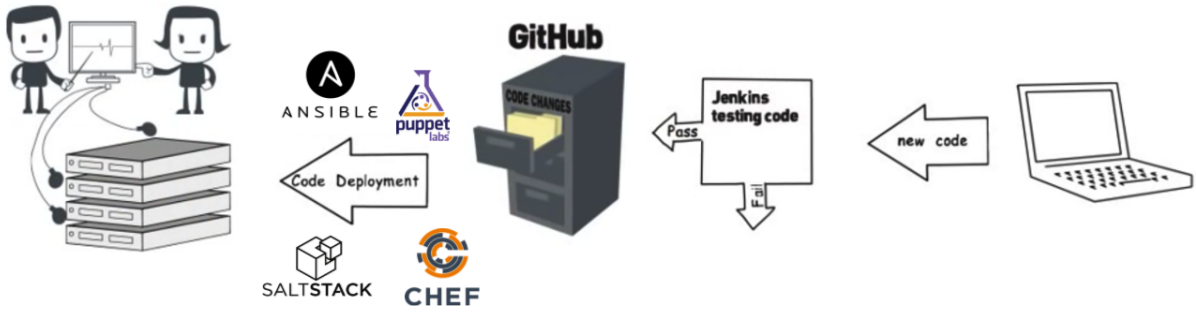
Jednou z hlavních výhod je jak již zmíněná rychlost nasazení nové verze. S tím je spojená i větší flexibilita společnosti reagovat na případné problémy a trendy na trhu. Zároveň i vývojáři získají pocit, že jejich práce má smysl, když mohou vidět nedávno vytvořené funkce využívané v reálném prostředí. (Amazon, 2017)

Další výhodou je jednoznačně spolehlivost a kvalita aplikace, kterou je možné velmi dobře měřit a dosáhnout tak co nejlepšího uživatelského prožitku koncového zákazníka. Monitorování a logování je možné sledovat v reálném čase a na základě těchto dat upravit aplikaci v řádech několika hodin. (Amazon, 2017)

V neposlední řadě je nutno zmínit i zlepšení kooperace jednotlivých týmů při používání DevOps. Sdílená zodpovědnost a vlastnictví klade důraz na společné hodnoty a spolupráci. Tím se zvyšuje efektivita práce, především díky odstranění čekání na reakci dalšího oddělení, které je v DevOps součástí jednoho celku. (Rackspace, 2013)

5.3. Nástroje DevOps

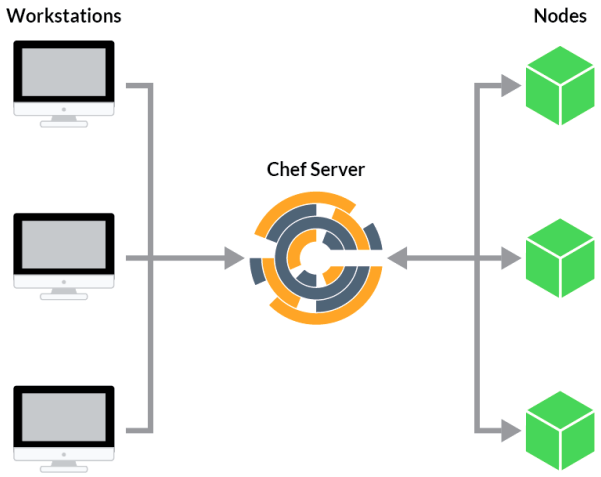
Proces začíná u vývojáře, který na své stanici napíše kód k nové verzi aplikace, ten je ovšem nutné automaticky nasadit a testovat současně. Příkladem takového nástroje může být Jenkins, v současnosti nejpoužívanější open source automatizovaný server. (Rackspace, 2013)



Obrázek 4 Využití nástrojů v DevOps (Zdroj:Rackspace, 2017)

Pokud je vše v pořádku, tak tento kód chce sdílet se zbytkem týmu, aby se tak jeho kód dostal do finální verze programu. K tomuto účelu se využívá verzovací služby, která nejen umožňuje kód sdílet, ale hlavně sledovat jeho změny a případně se vracet k předchozím úpravám. Příkladem takové služby je například GitHub. (Rackspace, 2013)
Nyní je zapotřebí nástrojů pro nasazení kódu přímo na server, které jsou automatizované a dokáží si poradit s různými konfiguracemi bez nutnosti ručního nasazení na jednotlivé servery. Jednou z takovýchto služeb je například Chef. (Chef, 2014)

5.3.1. Nástroj Chef

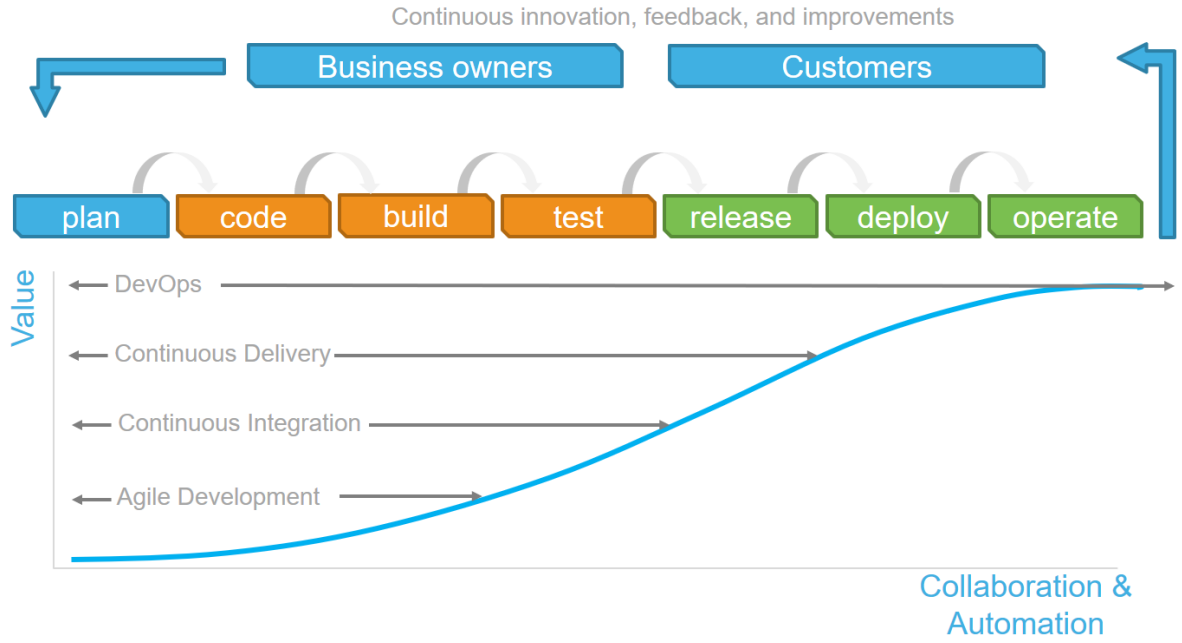


Obrázek 5 Schéma komunikace s Chef serverem (zdroj: Chef, 2017)

Chef je automatizační nástroj pro řízení a konfiguraci firemní infrastruktury fungující lokálně nebo i externě jako SaaS. Používá se primárně pro zrychlení nasazení softwaru, ale jeho možnosti tím zdaleka nekončí neboť dokáže vytvořit kompletní framework pro dosažení continuos delivery. Jádrem systému je Chef server, který ukládá konfigurace všech stanic, na kterých běží spravovaná aplikace (nodes). Tyto stanice se neustále dotazují Chef serveru na aktuální verzi a konfiguraci a pokud dojde ke změně, automaticky se aktualizují. Na druhé straně schématu stojí pracovní stanice (workstations), na kterých pracují vývojáři na novém kódu. V případě, že vývojář chce uvolnit novou verzi aplikace, pouze ji pošle na Chef server, sdělí mu, na jaké stanice jej chce poslat a Chef už zbytek obstará za něj. Stanice lze také dělit na skupiny a tvořit tak různá prostředí a oddělit např. produkční prostředí od vývojářského na kterém mohou vývojáři experimentovat podle libosti a zaměstnanci provozu mohou klidně spát. Toto řešení nabízí velkou flexibilitu a takřka neomezenou škálovatelnost. (Chef, 2014)

6. Spojení DevOps a agilního ALM

Ke správnému pochopení, jak může DevOps podpořit agilní ALM je nutné pochopit principy agilních metodik a zkombinovat je s osvědčenými přístupy DevOps. ALM poskytuje jakýsi framework pro uplatnění DevOps napříč celým životním cyklem, nikoliv jen vývojem a provozem. Snahou je uplatnit přístupy DevOps na všechny zúčastněné v životním cyklu a spolupracovat tak na vývoji co nejrychleji rostoucího a kvalitního softwaru.



Obrázek 6 Míry spolupráce a automatizace (Zdroj: Almbok, 2017)

Na obrázku výše je vidět že přidaná hodnota roste s tím, jak se zvyšuje poměr spolupráce a automatizace. U DevOps jsou tyto hodnoty nejvyšší, neboť pokrývá celý životní cyklus aplikace od plánování až po provoz a tím zaručuje rychlejší zpětnou vazbu od zákazníka, rychlejší formulaci nových business požadavků apod. (Yllemo, 2016; Yllemo, 2016)

6.1. Využití DevOps v agilním ALM

K dosažené průběžného vylepšování a integrace je možné využít agilního ALM, jako frameworku a následovat tyto 4 důležité kroky. (Aiello a Sachs, 2016)

6.1.1. Podpora spolupráce

Najít všechny stakeholdery projektu může být obtížné, natož udržet si přehled o každodenních činnostech členů týmu. K porozumění, co který tým na projektu dělá je třeba podporovat spolupráci napříč celým projektovým týmem. S velkými týmy přichází řada složitých vnitro týmových závislostí a DevOps pomáhá najít celkový pohled na tento systém a zorientovat se v jeho fungování. Toho je docíleno pomocí obohacení komunikace a týmové spolupráce, což je přesně oblastí, kde jsou DevOps praktiky a postupy nejvíce hodnotné. Tyto výhody dále podporují také automatizační nástroje a s jejich pomocí lze dosáhnout ještě vyšší přehlednosti a dostupnosti informací. (Aiello a Sachs, 2016)

6.1.2. Modelování pracovních postupů

Aby bylo možné dosáhnout úspěchu, je třeba diskutovat a modelovat postup práce na projektu se všemi zúčastněnými a dále využít automatizačních nástrojů pro sledování stavu rozdělených úkolů. Je běžné využívat Scrum meetingy k diskuzi o dokončené práci a případných problémech a překážkách, jež se objevily a vyžadují řešení. Automatizační nástroj může být využit k dopravení informací potřebných k dokončení práce a komunikaci s lidmi jež jsou za její dokončení zodpovědní. Častým problémem těchto nástrojů je jejich obtížné používání, a ještě obtížnější nastavení specifickým potřebám každého týmu. Je tedy velice důležité zamyslet se nad výběrem toho správného nástroje pro vlastní potřeby. (Aiello a Sachs, 2016)

6.1.3. Integrace procesů

Integrovat procesy vývoje a provozu je jedním z dalších zásadních faktorů. Procesy pro řízení změn by měli být co nejlehčí, přesto však schopné identifikovat jakákoliv technická rizika a podpořit jejich opravu.

V první řadě je nutné identifikovat procesy, které lze klasifikovat za rutinní, odstranit je z kontrolních meetingů a zaměřit se na procesy, které by mohli vyústit v případné problémy. Spolu s tím je velmi efektivní integrovat incident a problem managment procesy, aby fungovali jako zpětná vazba a zároveň neustále zlepšovat procesy vývoje. (Aiello a Sachs, 2016)

Je nutné mít dostatečné množství procesů pro odvrácení chyb, ale ne příliš, aby nebyla zavedena přílišná byrokracie a aby nedocházelo k omezování pracovníků. (Aiello a Sachs, 2016)

6.1.4. Dodržování požadavků

Agilní ALM by mělo pomoci dodržovat regulace a požadavky auditu a zároveň vytvořit dostatečnou zpětnou vazbu pro top management k usnadnění IT governance a rozhodování. K tomu je zapotřebí efektivních požadavků, pro které je nutné nejdříve pochopit principy a praktiky agilního přístupu. Hlavní myšlenkou a snahou tohoto přístupu je tvořit kvalitu od samého začátku a v tomto duchu by se měli nést i požadavky. (Aiello a Sachs 2016)
Na začátku vývoje je nejlepší začít s minimem procesů a průběžně přidávat blíže k dni vydání. Na začátku projektu totiž není možné mít úplně jasnou představu o všech aspektech finálního produktu a není tak možné definovat ihned všechny požadavky. Právě ruku v ruce se stálým vylepšováním softwaru jde i stále zlepšování procesů, což je tou nejefektivnější cestou, která pomůže vyhnout se drahým chybám a časově náročným úpravám. (Aiello a Sachs, 2016)

7. Závěr

DevOps již není jen o spojení vývoje a provozu, ale je možné tyto principy s využitím agilního ALM aplikovat na celý životní cyklus softwaru. Hlavním cílem tohoto spojení je pomoci společnostem tvořit kvalitní software v co nejkratším čase a postupně jej zlepšovat, monitorovat a automatizovat. Celé toto spojení stojí na komunikaci mezi týmy a společnou snahou zvýšit produktivitu a zajistit maximální spolehlivost.

Dva hlavní body, které je nutné si uvědomit je, že agilní ALM zvyšuje rychlost a DevOps zajišťuje, že celková snaha je na správném místě a je tak možné dosáhnout co nejproduktivnějšího prostředí. (Aiello a Sachs, 2016)

8. Zdroje

Aiello, Bob a Leslie Sachs, 2016. Use DevOps to Drive Your Agile ALM.
Aiello Bob a Leslie Sachs. 2014. Drive agile Application Lifecycle Management with DevOps. IBM - United States. [Online] 3. 4 2014. [Citace: 10. 5 2017.] Dostupné z: https://www.ibm.com/developerworks/library/d-drive-agile-lifecycle-management-devops/.
Almbok, 2017. Application Lifecycle Management *[ALM]* [online] [vid. 2017-05-14]. Dostupné z: http://www.almbok.com/
Appelo, Jurgen. 2010. Agile Application Lifecycle Management (ALM). SlideShare. [Online] 22. 10 2010. [Citace: 13. 5 2017.] Dostupně z: https://www.slideshare.net/jurgenappelo/agile-alm.
Chef. 2014. Intro to Chef. Youtube. [Online] 21. 5 2014. [Citace: 10. 5 2017.] Dostupné z: https://www.youtube.com/watch?v=jlwGcgFfcnU&t=5s.
Gene, Kim, 2013. DevOps distilled [online] [vid. 2017-03-11]. Dostupné z: http://www.ibm.com/developerworks/library/se-devops/part1/index.html
Harvey, Cynthia, 2017. 10 Ways DevOps Is Changing Enterprise IT - Datamation [online] [vid. 2017-03-11]. Dostupné z: http://www.datamation.com/data-center/slideshows/10-ways-devops-is-changing-enterprise-it.html
Loukides, Mike, nedatováno. What is DevOps? - O'Reilly Radar [online] [vid. 2017-03-11]. Dostupné z: http://radar.oreilly.com/2012/06/what-is-devops.html
Owen, Gerie. 2017. Choosing the right ALM software for your organization. Tech Target. [Online] 2017. [Citace: 12. 5 2017.] Dostupné z: http://searchsoftwarequality.techtarget.com/feature/Choosing-the-right-ALM-software-for-your-organization.
Yllemo, Henrik. 2016. Welcome [ALM[. [Online] 16. 12 2016. [Citace: 11. 5 2017.] Dostupné z: http://www.almbok.com/.
—. 2016. ALM - Agile - DevOps. Youtube. [Online] 8. 9 2016. [Citace: 10. 5 2017.] Dostupné z: https://www.youtube.com/watch?v=plzJFh76DJc.