Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Výběr a nasazení řešení pro řízení vývoje softwarových projektů

Bc. Patrik Bláha

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Patrik Bláha

Informatika

Název práce

Výběr a nasazení řešení pro řízení vývoje softwarových projektů

Název anglicky

The selection and deployment of solutions for managing software development

---

Cíle práce

Hlavním cílem práce je pomocí vícekriteriálních metod porovnat v současnosti dostupná řešení pro řízení vývoje softwarových projektů a navrhnout konkrétní řešení pro zvolený podnikatelský subjekt. Pro dosažení hlavního cíle bude využito získaných poznatků ze sekundárních zdrojů.

Metodika

Na základě studia odborné a vědecké literatury a provedené analýzy sekundárních zdrojů, bude vytvořena syntéza poznatků. Budou zvoleny vhodné metody zpracování a bude navržen postup řešení praktické části diplomové práce.

Na základě získaných informací budou stanovena jednotlivá kritéria hodnocení pro porovnání dostupných řešení, bude provedeno vlastní porovnání dostupných řešení pomocí vícekriteriální metody. V praktické části bude navrženo vhodné řešení pro konkrétní subjekt aplikací vah jednotlivých kritérií vícekriteriální porovnávací metody.

Doporučený rozsah práce

60 stran

Klíčová slova

řízení IT projektů, cloud, DevOps, vícekriteriální porovnávací metody

---

Doporučené zdroje informací

JAMSA, Kris. Cloud computing: SaaS, PaaS, IaaS, virtualization, business models, mobile, security and more. Burlington, MA: Jones & Bartlett Learning, c2013. ISBN 9781449647391.

KORT, Wouter de. DevOps on the Microsoft stack. New York, NY: Apress, [2016]. ISBN 978-1-4842-1447-3.

Ochrana osobních údajů: zákon o ochraně osobních údajů a další právní předpisy. GDPR – obecné nařízení Evropského parlamentu a rady (EU) 2016/679, o ochraně osobních údajů : redakční uzávěrka 28.8.2017. Ostrava: Sagit, [2017]. ÚZ. ISBN 978-80-7488-241-8.

VELTE, Anthony T, Toby J VELTE a Robert C ELSENPETER. Cloud Computing: praktický průvodce. Brno: Computer Press, 2011. ISBN 9788025133330.

WEBBER-CROSS, Geoff. Learning Microsoft Azure. Birmingham – Mumbai: Packt Publishing, 2014. ISBN 9781782173373.

Předběžný termín obhajoby

2019/20 ZS – PEF (únor 2020)

Vedoucí práce

Ing. Edita Šilerová, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 19. 10. 2018

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 19. 10. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 16. 11. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Výběr a nasazení řešení pro řízení vývoje softwarových projektů“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 16. 11. 2019 _____

Poděkování

Rád bych touto cestou poděkoval vedoucí práce Ing. Editě Šilerové, Ph.D. za odborný dohled a poskytnuté konzultace.

Výběr a nasazení řešení pro řízení vývoje softwarových projektů

Abstrakt

Současným trendem ve vývoji software je minimalizace technologického dluhu a automatizace procesů pomocí relativně mladé metodologie označované zkratkou DevOps, která zastřešuje životní cyklus softwarového projektu včetně jeho plánování, realizace, testování a nasazení, ale i provozování a reakce na zpětnou vazbu.

Na vývoj software nahlíží agilním pohledem jako na nikdy nekončící proces postupného zdokonalování pomocí velmi rychlé integrace a nasazování velkého množství drobných změn. Tento přístup má na první pohled samé výhody – zákazník bude spokojený s rychlou implementací jeho požadavků, vývojáři se značně usnadní život díky automatizaci neproduktivních činností, software bude bezpečnější díky automatizovanému testování a uživatelé se dočkají skvělé podpory při řešení komplikací. Adopce DevOps je však komplikovaný úkol vzhledem k nutnosti využít při realizaci velké množství nástrojů, jejichž orchestrace je netriviálním úkolem, který trápí velké i malé vývojářské týmy. Roztříštěnost je zde brzdou pokroku.

Tento problém se výrobci vývojářských nástrojů rozhodli řešit vytvářením ekosystémů vzájemně dobře propojených a vyladěných nástrojů, které mají ambici postihnout metodologii DevOps v celé její šíři. Kvůli značné komplexitě však tyto ekosystémy mezi sebou nejsou jednoduše porovnatelné, což znesnadňuje vývojářským subjektům volbu řešení, které by odpovídalo jejich individuálním potřebám. Předmětem této práce je navrhnout model porovnání těchto ekosystému pomocí vhodné vícekriteriální metody a demonstrovat jeho použití na konkrétním vývojářském subjektu.

Klíčová slova: řízení IT projektů, cloud, DevOps, vícekriteriální porovnávací metody

The selection and deployment of solutions for managing software development

Abstract

The current trend in software development is to minimize technological debt and automate processes using a relatively recent methodology called the DevOps, which covers the software project lifecycle including its planning, implementation, testing and deployment, as well as running and responding to feedback.

It looks at software development with an agile perspective and sees it as a never-ending process of gradual improvements through very fast integration of large number of small changes. At first glance, this approach has all the benefits - the customers will be satisfied with the rapid implementation of their requirements, developer's life will be much easier with automations of non-productive activities, the software will be more secured through automated testing, and users will receive great support with issue resolution. Adoption of DevOps is however a complicated task due to the necessity of using many instruments whose orchestration is a non-trivial task that poses a challenge for large and small development teams alike. Fragmentation hinders progress.

Makers of developer tools have decided to address this issue by creating ecosystems of well-interconnected and fine-tuned tools which have the ambition to cover the DevOps methodology in its entirety. However, because of its complexity, these ecosystems are not easily comparable, making it difficult for developers to choose a solution that fits their individual needs. The goal of this thesis is to propose a model for comparing these ecosystems using a suitable multi-criteria method and to demonstrate its use on a selected software company.

Keywords: project management, cloud, DevOps, multicriteria comparison methods

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Vývoj software, stejně jako jakákoliv jiná komerční aktivita, tvoří soubor činností a vzájemně souvisejících procesů, které je nutné určitým způsobem řídit, aby byla tato aktivita úspěšná. Pro tento úkol se používají různé nástroje, techniky a systémy. Naivní představa začínajícího programátora by se mohla omezovat pouze na nástroje pro správu a verzování zdrojového kódu. Problematika je ve skutečnosti mnohem složitější, zejména pokud chápeme vývoj ze širšího pohledu jako kontinuální proces, který zahrnuje také testování, nasazení a provoz vyvíjeného software, včetně monitorování a optimalizace jeho provozu a zpracování zpětné vazby (hlášení chyb, feature-requests atd.) od jeho uživatelů, přičemž pro každý z těchto úkolů je nutné použít vhodný nástroj. Jelikož jde o činnost převážně týmovou, vyvstává rovněž potřeba organizace spolupráce jednotlivých osob, které na sebe berou různé týmové role, které poté realizují.

Výběr vhodné metodiky a posléze i potřebných nástrojů je klíčový nejen pro úspěch projektu v rovině technologické, ale také v rovině hospodářské. Z toho důvodu je třeba při jejich volbě vzít v úvahu také ekonomické faktory, zejména pak cenu a podmínky pro možnost budoucího škálování. Promyšleným výběrem komplexního řešení se lze vyhnout nákladným problémům v budoucnu.

Technologické a ekonomické faktory jsou přirozeně vzájemně závislé, proto je třeba provádět porovnání dostupných nástrojů z jejich hlediska simultánně. Cílem této práce je provést právě takové porovnání pomocí vícekriteriálních metod a demonstrovat jejich použití pro výběr řešení pro konkrétní vývojářský subjekt.

2. Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je pomocí vícekriteriálních metod porovnat v současnosti dostupná řešení pro řízení vývoje softwarových projektů a navrhnout konkrétní řešení pro zvolený podnikatelský subjekt. Pro dosažení hlavního cíle jsou stanoveny cíle dílčí.

Prvním dílčím cílem je získat poznatky ze sekundárních zdrojů a definovat teoretický rámec pro další práci jednak v rovině metodik, potřeb a ucelených řešení v současnosti používaných při vývoji software, jednak v rovině dostupných vícekriteriálních metod pro porovnání těchto řešení, včetně volby vhodné metody a definice konkrétních kritérií.

Druhým dílčím cílem je na základě tohoto teoretického rámce vybrat konkrétní v současnosti dostupná řešení jako kandidáty pro porovnání a extrahovat a normalizovat hodnoty zvolených kritérií na základě vlastností těchto konkrétních řešení.

Třetím dílčím cílem je syntéza poznatků z předchozích dílčích cílů, jejich aplikace na konkrétní subjekt, provedení vlastního porovnání a vyhodnocení výsledků.

2.2 Metodika

Na základě studia odborné a vědecké literatury a provedené analýzy sekundárních zdrojů, bude vytvořena syntéza poznatků. Budou zvoleny vhodné metody zpracování a bude navržen postup řešení praktické části diplomové práce.

Na základě získaných informací budou stanovena jednotlivá kritéria hodnocení pro porovnání dostupných řešení, bude provedeno vlastní porovnání dostupných řešení pomocí zvolené vícekriteriální metody. V praktické části bude navrženo vhodné řešení pro konkrétní subjekt aplikací vah jednotlivých kritérií vícekriteriální porovnávací metody.

3. Teoretická východiska

Tato kapitola stručně popisuje metodiky a nástroje používané při vývoji software a potřebný teoretický aparát týkající se výzkumných metod pro provedení porovnání těchto nástrojů, na jehož základě jsou formulovány závěry dále v této práci.

3.1 Metodiky používané pro vývoj software

Protože oblast vývoje softwaru prošla určitým historickým vývojem, existuje v současné době pestrá paleta paradigmat, modelů a k nim příslušejících metodik, které mohou vývojové týmy adoptovat. Tyto modely a metodiky můžeme obecně rozdělit na tradiční (rigidní, klasické, s hierarchickou strukturou řízení) a modernější agilní (dynamické, iterativní, s plochou a samo-organizující se strukturou řízení).

Mezi tradiční metodiky řadíme zejména přístupy typu Vodopád nebo RUP (Rational Unified Process), které kladou velký důraz na detailní analýzu, návrh, plánování a dokumentaci. Činnost jednotlivých vývojářů je velmi jasně vymezena jejich pozicí v hierarchické personální struktuře. Veškeré procesy jsou velmi formálně popisovány, zejména s pomocí jazyka UML.

Agilní metodiky naproti tomu kladou důraz na samotný výsledek vývoje a snaží se oprostit od přebujelé dokumentace a interní byrokracie spojené s rigidními postupy tradičních metodik. Vycházejí z premisy, že vývoj softwaru je dynamický a kontinuální proces, při kterém se požadavky zákazníka mohou měnit, a to i v pozdních stádiích vývoje projektu. Z toho důvodu podporují pružnou spolupráci v samo-organizujících se týmech a nabádají k průběžnému a nasazování výsledku práce týmu v malých rychloobrátkových iteracích. Více viz Manifesto for Agile Software Development [8]. Mezi nejpoužívanější metodiky této kategorie patří SCRUM, FDD (feature driven development) a XP (Extreme programming). Díky omezení nadbytečných činností a schopnosti pružně a rychle reagovat na změny se agilní metodiky stávají stále více oblíbené [9][10], a to i ve velkých korporátních společnostech jako je např. Microsoft [11].

Tomuto trendu, který bude zřejmě pokračovat i v následujících letech [12], se přirozeně přizpůsobují také nástroje pro řízení softwarových projektů, které prochází postupnou integrací jednotlivých komponent do komplexních, většinou cloudových, řešení.

3.2 DevOps

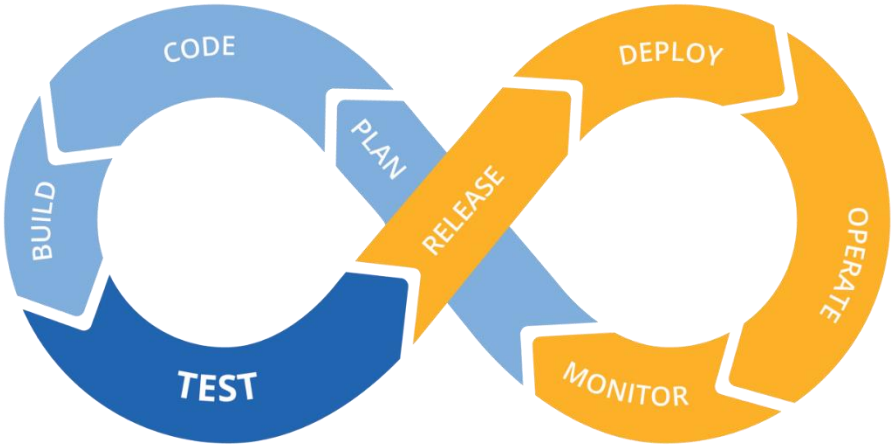
V souvislosti s těmito změnami je v posledních letech stále více zmiňován termín DevOps, který představuje moderní metodologii vývoje software vzniklou v reakci na zvyšující se dynamiku tohoto odvětví. Jelikož jde o velmi mladý fenomén (dle dostupných zdrojů se tato myšlenka objevila r. 2010 [13]), není jeho definice, zejména v akademické sféře, dostatečně ustálena. Poměrně abstraktní definici navrhli LEN BASS, INGO WEBER, a LIMING ZHU z Carnegie Mellon Software Engineering Institute, a to:

> *„DevOps vyjadřuje soubor postupů určených ke zkrácení doby mezi uskutečněním změny systému a nasazením této změny do běžné produkce při zajištění vysoké kvality“ [14] (přeloženo autorem).*

DevOps bývá někdy nadneseně označován rovnou za kulturu či filosofii [7], v kontextu této práce jsou sledovány pouze praktické aspekty a benefity této techniky, protože ačkoliv se tato práce nezabývá přímo DevOps jako takovým, poslouží díky své všeobjímající povaze pro identifikaci kategorií nástrojů, na které se později zaměří stanovování obecných kritérií pro porovnání dostupných konkrétních řešení.

Na přístupu DevOps je zajímavá a inovativní zejména změna úhlu pohledu na vývoj jako takový. Termín DevOps je složeninou výrazů Development (vývoj) a Operations (provoz). DevOps vychází z myšlenky, že vývoj softwaru se nesestává jen z vytváření zdrojového kódu (*code*) organizovaného pomocí některé agilní metodiky (*plan*), ale zahrnuje také všechny další „podpůrné“ činnosti jako jeho nedílnou součást s tím, že jejich výstupy mají sloužit jako podklad pro další směřování vývoje. Tím je myšleno zejména testování kódu (*test*), sestavování artefaktů (*build*), správa verzí vydání (*release*), jejich nasazení (*deploy*), konfigurace (*operate*) a monitoring (*monitor*) používání aplikace koncovým zákazníkem.

Smysl DevOps leží zejména v propojování těchto dříve oddělených procesů do jednoho pomyslného řetězce (označovaný jako tzv. *lifecycle*), kdy výstupy jednoho procesu tvoří vstup následujícího, přičemž celý řetězec se na konci uzavírá a tvoří tak nekonečnou smyčku představující vývoji softwaru jako nekončící, interaktivní proces, viz Obrázek 1.



Obrázek 1: DevOps lifecycle, zdroj: [15]

DevOps podporuje úzké propojení mezi jednotlivými fázemi vývoje a provozu pomocí co nejvyšší míry automatizace procesů. V souvislosti s tím se do popředí dostávají praktiky průběžné integrace / nasazování (*Continuous Integration / Continuous delivery,* též *CI/CD*), které provádí různé úlohy (testování, sestavování, nasazování) na základě definovaných pravidel nad určitou dynamickou strukturou, zejména pak nad větvemi repozitářů zdrojového kódu. Tato automatizace bývá obvykle založena na zpracování automaticky generovaných událostí mezi jednotlivými součástmi stylem *If This, Than That* (KDYŽ … PAK …), např. Když proběhne sloučení (*merge*) do hlavní větve repozitáře, PAK vezmi tento kód a proveď na něm testy. Když tyto testy proběhnou v pořádku, PAK proveď sestavení… a tak dále. Pravidla se nemusí omezit pouze na události související se správou kódu, ale lze do nich zahrnout i události týkající se provozu (např. provozní chyby), zpětné vazby (hlášení chyb, reporting), nebo řízení projektu (schvalovací procesy).

Kolekce všech takových automatizovaných pravidel dohromady tvoří workflow, které může pokrývat všechny aspekty celého procesu vývoje a provozování daného softwarového projektu. Při vhodném nastavení a zapojení schvalovacích procesů na vhodná místa, může CI/CD minimalizovat tzv. technických dluh (*technical debt*) vývojového procesu a tím ho zrychlit. Každý úkol, který má být automatizován, je typicky prováděn pomocí jiného nástroje.

K provázání těchto nástrojů se používají tzv. pipelines (doslova „roury“), které představují datové toky mezi jednotlivými nástroji. Z praktického pohledu DevOps nepředepisuje žádné konkrétní nástroje, které mají být použity, klade pouze důraz na jejich dobrou provázanost pro hladký chod celého vývojového procesu.

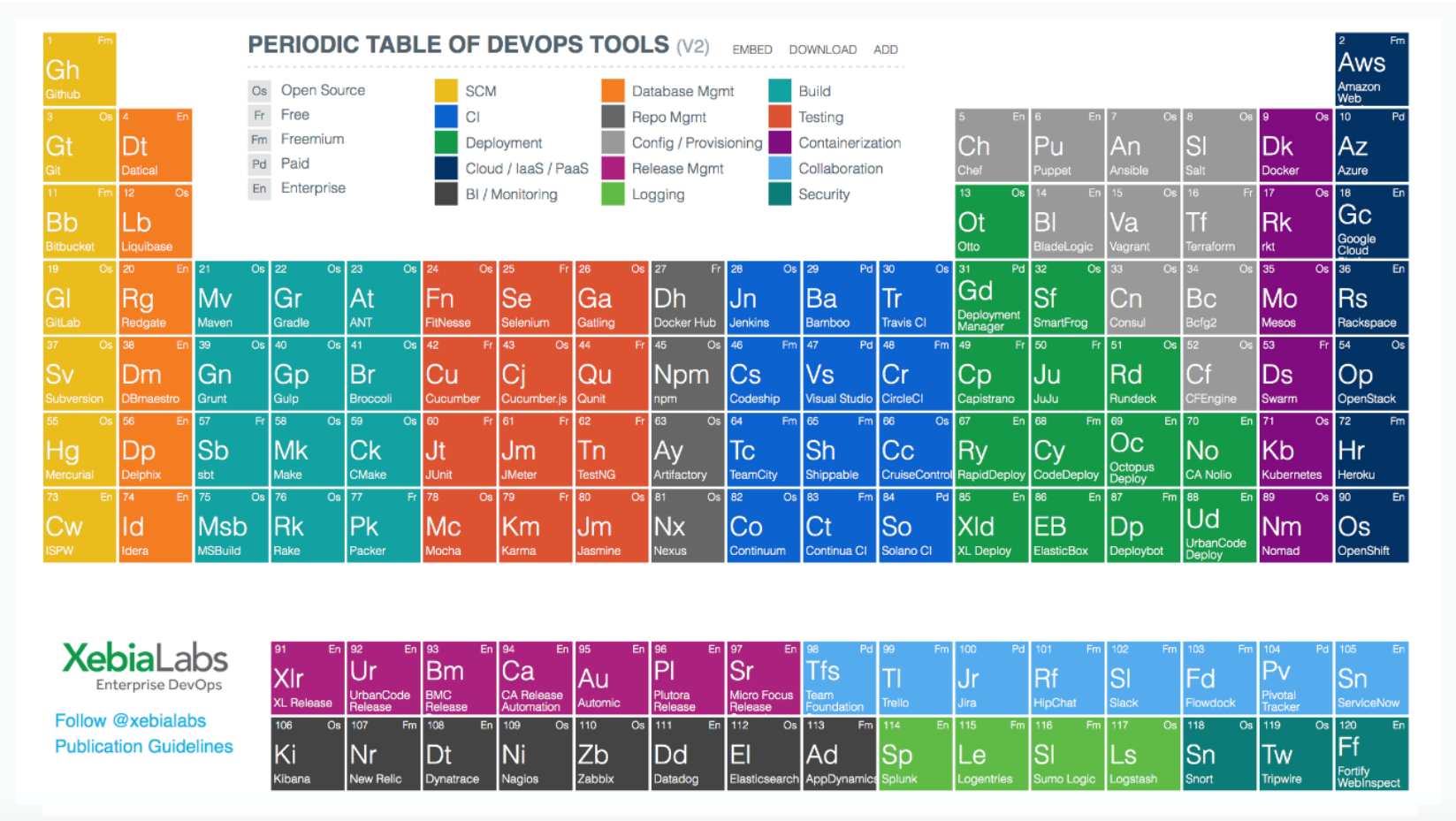
3.3 Kategorie nástrojů používaných pro vývoj a jejich charakteristiky

Nástroje používané pro vývoj spadají do různých kategorií, které se vzájemně prostupují, a jejich kategorizace je tak komplexní. Standardní předepsané procesy, které se prolínají napříč DevOps workflow a byly osvojeny jednotlivými organizacemi napříč světem byly rozděleny do následujících obecných kategorií [17]:

- Nástroje pro plánování a spolupráci
- Správa zdrojového kódu
- Nástroje pro sestavování (build)
- CI / CD nástroje
- Automatické testování
- Podpora a zpětná vazba

Ve skutečnosti tato základní kategorizace popisuje širokou škálu nástrojů, jež nejlépe popisuje obrázek 2 s tabulkou níže. Tato tabulka čítá na 119 různých softwarových nástrojů, rozdělených podle praktických úkolů, které tyto základní kategorie obstarávají.

Generalizace obecných kategorií napomáhá v adaptaci na používání jednotlivých softwarových částí k zaručení dosažení klíčových cílů vedoucích k úspěchu [17].



Obrázek 2: "Periodická tabulka" nástrojů DevOps od XebiaLabs, zdroj: [16]

3.3.1 Nástroje pro plánování a spolupráci

Základní obecnou kategorií nástrojů DevOps jsou nástroje pro plánování a spolupráci. Právě tato kategorie, ačkoliv se zdá být na první pohled nejméně užitečná, sebou nese jeden z nejzákladnějších pilířů důležitých právě pro vývoj jakéhokoliv softwarového produktu. Často opomíjena vývojáři, právě pro to, že při vývoji přispívá spíše teoretickou, nelži praktickou částí. Je využívána zejména managementem pro řízení, správu a informování o stavu úkolů (často používané slovo z anglického jazyka „ticketů“ či „tásků“).

Vzhledem k neustále rostoucímu trendu, kdy jsou agilní metodiky stále více oblíbené a využívané i velkými korporátními společnostmi [9][10][11], se i většina softwarových produktů zařazených pod touto kategorií vztahují právě k těmto metodikám.

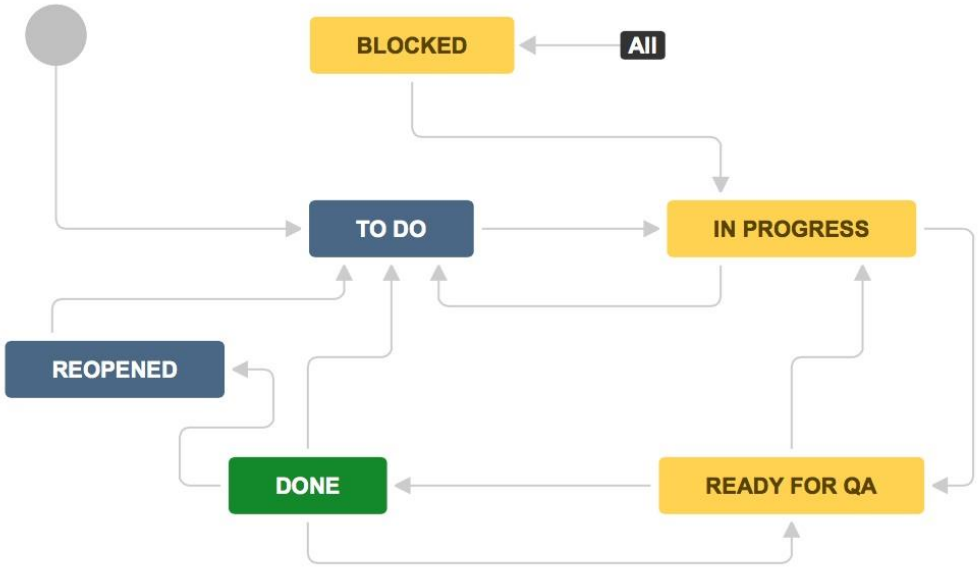
Tyto nástroje primárně slouží pro plánování a řízení zadané práce, tedy úkolů, na kterých pracuje celý tým. Každý nástroj může být do jisté míry odlišný, avšak co se funkčnosti týká, měl by obsahovat podporu pro řízení agilního vývoje. Mezi tyto základní funkce paří zejména vytváření úkolů a jejich správu, jejich vzájemné provázání (vzhledem k projektovému řízení), možnost odhadu doby trvání úkolu (v případě agilní metodiky SCRUM). Dále také správu obsahu úkolu (popis, dodatečné informace), správu souborových příloh, informace o tom, kdo úkol zadal, kdo ho má na starosti. Mimo jiné by měl obsahovat i kompletní auditní log, aby bylo přehledně vidět, kdo, kdy a jakou akci na úkolu vytvořil [7].

Tyto nástroje, s přihlédnutím k agilním metodikám, umí v obecném měřítku spravovat workflow těchto úkolů. V případě agilní metodiky SCRUM, která je nejvíce oblíbenou agilní metodikou [18], lze definovat základní stavy, které se prolínají napříč celým workflow:

- Backlog – tvoří zásobník úkolů
- To do – vybráno pro implementaci, čeká na zvolení
- In progress – práce právě probíhá
- Blocked – blokován
- Reopend – znovu otevřen
- Done – hotovo
- Ready for QA – připraven k testování

V metodice SCRUM tvoří Backlog základní zásobník úkolů. Tyto úkoly mohou být rozděleny na jednotlivá „User stories“. User story v překladu znamená příběh uživatele, tedy obecný popis situace, která je popsána uživatelem. Ta pod sebou může dále mít jednotlivé miniaturní úkoly (anglicky subtasks), které vedou k realizaci celé user story [18]. SCRUM je dále rozdělen na jednotlivé iterace, které se anglicky nazývají „Sprints“.

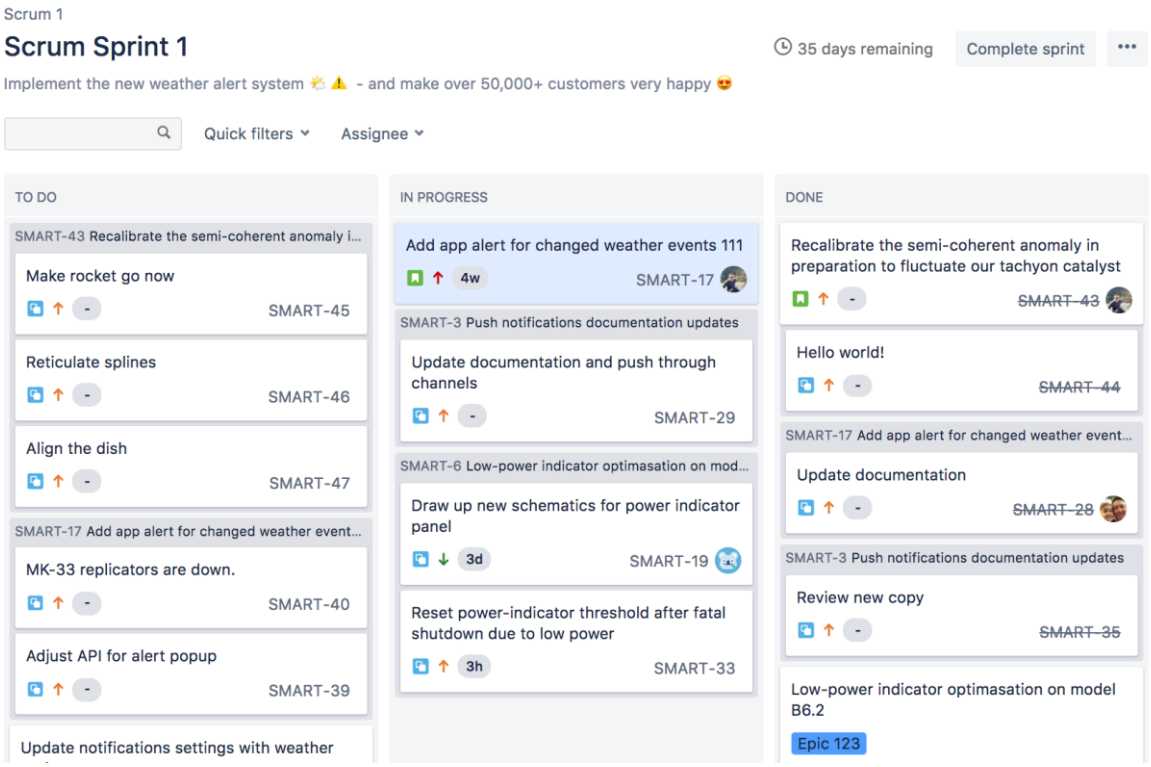
Při nastartování každého nového sprintu, který běžně trvá zhruba 10 dní přejdou vybrané úkoly z backlogu do stavu „To do“. Tímto jsou vybrány k implementaci a zařazují se tak do definovaného workflow. V momentu, kdy jsou přiřazené pro vývoj, přecházejí do stavu „In progress“, ten indikuje právě probíhající práci na zadaném úkolu. V případě, že je potřeba dodefinovat k úkolu více informací, nebo je například blokován jiným úkolem, přechází do stavu „Blocked“. V případě že je úkol hotový, přechází do stavu „Ready for QA“ a následně je testován. Je-li při testování zjištěno, že není hotový, přechází do stavu „Reopend“ a prochází tak celý proces znovu. V opačném případě končí ve stavu „Done“, tedy hotovo a je uzavřen. Daná „User Story“ přechází do stavu „Done“ v případě že jsou všechny její úkoly hotové [19]. Celá tato definice worflow je uvedena na následujícím obrázku.



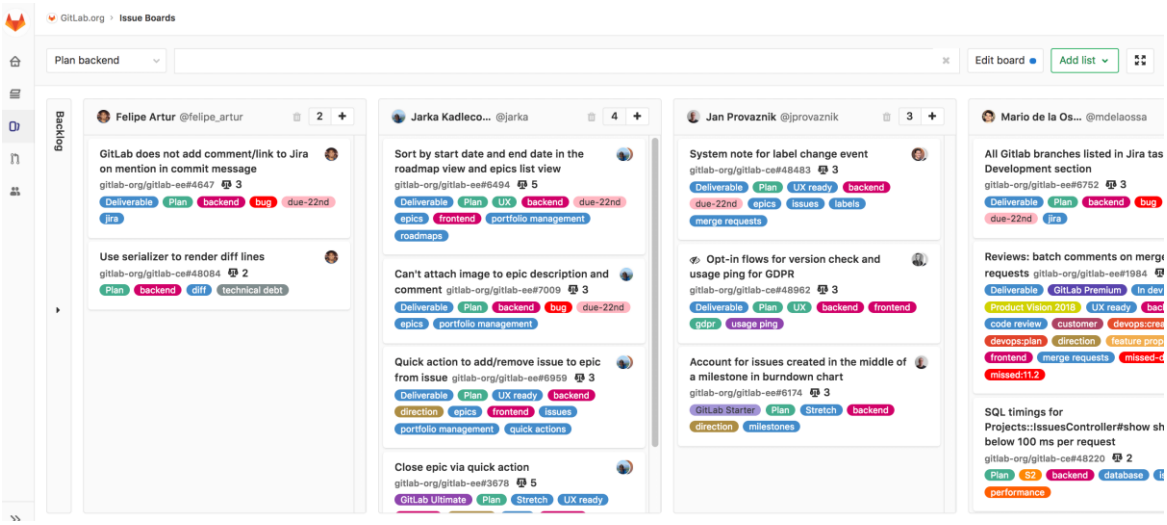
Obrázek 3: Atlassian SCRUM worklow, zdroj: [19]

Toto je pouze zjednodušený popis metodiky SCRUM, která samozřejmě může být více komplexní, avšak díky dynamickému workflow, které umožňuje většina nástrojů pro spolupráci nastavit, může být toto vytvořeno dle potřeb koncového uživatele. Vhodnými kritérii na nástroje řešící tuto problematiku tedy budou zejména SCRUM, Kanban a agile reporting [19].

Mezi konkrétní zástupce lze řadit například nástroj Jira z ekosystému firmy Atlassian [28], nebo software GitLab [29] od stejnojmenné společnosti. Na následujících obrázcích jsou uvedeny ukázky jednotlivých nástrojů a jejich „boardů“.



Obrázek 4: Atlassian SCRUM board, zdroj: [7]



Obrázek 5: Gitlab SCRUM board, zdroj: [20]

3.3.2 Správa zdrojového kódu

SCM – Source Code Management, neboli správa zdrojového kódu, má na starost ukládání zdrojového kódu a jeho správu. Nabízí velkou škálu možností a benefitů. Správa zdrojového kódu dovoluje vícero vývojářům pracovat na stejném kódu aplikace v různý čas, v různých vývojářských centrech, a to kdekoli na zemi. Je využívána již řadu let a v poslední době se stala jednou z integrací celého ekosystému DevOps.

SCM má na starosti verzování kódu a pomáhá držet základní strukturu, či podstrukturu verzí kódu. V principu může fungovat správa větví na základě definovaného toku (angl. flow). Pro tyto flow existují tzv. best practises. Jedním z nich je například git flow, viz [50]. Při využití předem definovaného git flow může být vytvořena hlavní větev, takzvaná „master“ větev, která drží aktuální podobu kódu spuštěného nad produkčním prostředím a větev „develop“, která drží verzi kódu spuštěného na neprodukčním, tedy testovacím, nebo také vývojovém prostředí [17].

Každý vývojář si posléze může vytvořit vlastní větvě, ve kterých si drží požadované změny a tyto větve se následně spojují (z anglického slova „merge“) dohromady. Dále je také umožněno vytváření tzv. „bug fixing“ větví, které se starají o opravu chyb v hlavních větví. Díky tomuto principu je umožněno paralelního vývoje.

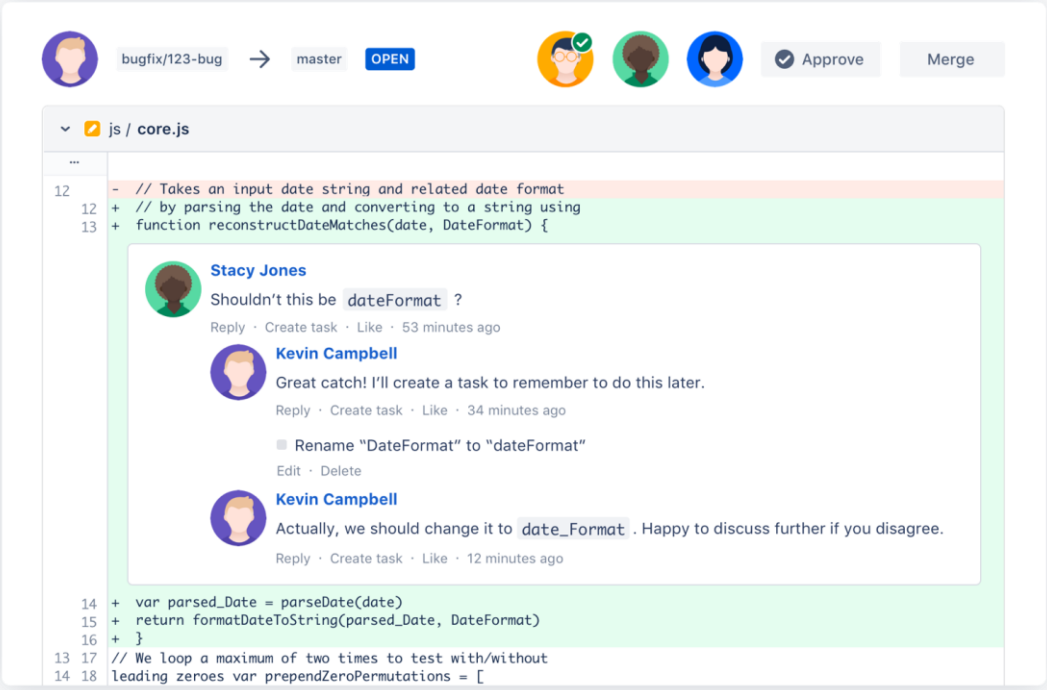
Všechny tyto větvě obsahují auditní log, sledování změn a celou historii vývoje. Ta je následně po spojení větví přenesena do hlavní větvě. Tento princip zaručuje transparentní správu všech prostředích potřebných pro testování, vývoj a produkci.

Správa zdrojového kódu doplňuje proces DevOps o tyto základní prvky [17]:

- Jednoznačná definice pravdy pro celý vývojový tým
- Revize změn před spojením s hlavní větví
- Spolupráce, spoluautorství a jednotlivé změny
- Audit změn s možností vrácení změn
- Inkrementální záloha

Spolu s tímto je dobré zmínit, že SCM zabezpečuje i takzvané „Code Review“, což je procesní možnost, díky které je sepsaný kód revidován jiným vývojářem a popřípadě komentován viz následující obrázek, nebo vrácen k přepsání. V případě úspěšného „Code Review“ je kód přijat a zařazen do dané větve [17].

Na základě získaných teoretických poznatků o správě zdrojového kódu budou jako vhodná kritéria sledována zejména, zda daný ekosystém obsahuje správu zdrojového kódu. Dalším vhodným kritériem bude sledováno, zda daný repozitář umožňuje emailové notifikace, revize zdrojového kódu, vkládání komentářů, akceptace merge požadavků a velikost úložiště.



Obrázek 6: Kontrola zdrojového kódu – Atlassian Bitbucket, zdroj: [21]

3.3.3 Nástroje pro sestavování (build)

Build neboli sestavování aplikace, není pouhým aktem, při kterém se ze zdrojového kódu aplikace stává plně funkční celek. Tato akce, ačkoliv by se na první pohled mohlo zdát, že je velmi jednoduchá, je ve skutečnosti o něco více komplexní [22].

Samotný build jako takový v sobě může zahrnovat více aktivit, které s výsledným funkčním celkem aplikace souvisí. Jedná se zde například o testování při sestavování aplikace, její samostatné nasazení a v základu samozřejmě také kompilace. V reálném použití, si lze takovýto build představit jako proces, který s sebou nese jednotlivé dílčí části, které se musejí aplikovat před tím, než se úspěšně dokončí [22].

Jako jeden z takových to typických scénářů může vypadat například takto:

1. V první řadě vytvoří vývojář aplikace změnu, kterou následně zašle do repozitáře. Ten má za úkol správu kódu.
2. Mezi tímto prvním úkonem, zde existuje tzv. sestavovací (build) server, který si každou minutu stahuje aktuální repozitář a sleduje, zda se zdrojový kód změnil. Takovýto server lze považovat za Continuous Integration (CI) server, který bude popsán v následující kapitole.
3. Brzo po zaslání změny vývojáře tento server zjistí, že se zdrojový kód změnil a spustí nad ním tzv. build script, který následně software integruje.
4. Continuous Integration server zašle vybrané skupině vývojářů emailovou notifikaci s výsledkem ze spuštění sestavovacího skriptu.
5. Server nadále monitoruje, zdali se zdrojový kód změnil.

K vytvoření sestavovacího skriptu se využívá buď nástrojů používaného IDE nebo je konstruován ručně. Sestavovací skript je následně předán sestavovacímu nástroji, který podle něj provede požadované akce. Toto zpracování může probíhat synchronně i asynchronně.

Vhodným kritériem, které zde lze sledovat, je, kolik takovýchto skriptů lze paralelně spouštět, zda je výkonnost skriptu z nějakých důvodů omezená (např. čas běhu, přidělené prostředky – paměť, CPU atd.). Ve většině případů jsou kritéria v těchto případech omezená počtem spuštěných úloh a časem, který je přidělen k běhu. Toto se ale týká pouze cloudových řešení. On-premise řešení, které pro spouštění takovýchto úloh

využívá hardware, jenž daná instituce využívá, jsou limitovány výkonností jejich vlastních hardware [7].

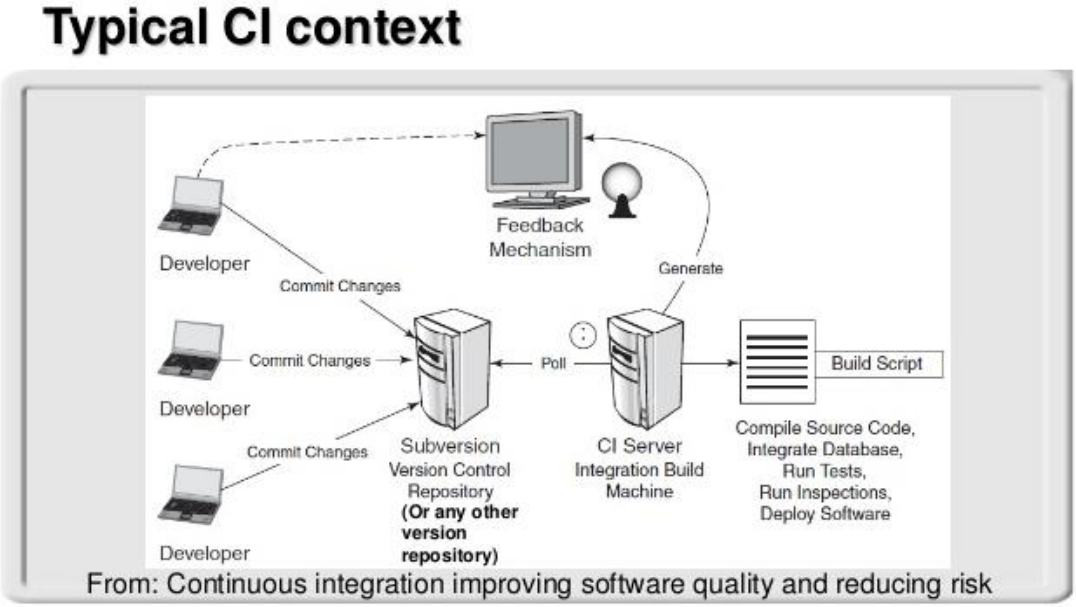
Dále bude také sledování, kolik místa úložného prostoru může být využito pro takzvané ukládání artefaktů. Artefakty jsou označovány jako výsledné produkty, vyprodukované sestavením zdrojových kódů [30].

3.3.4 CI/CD nástroje

Continuous integration a continuous delivery jsou činnosti sloužící k usnadnění práce při průběžném nasazování změn zdrojového kódu. Jedním z velkých pozitiv, které spolu s sebou přinášejí, je fakt, že k nasazování, testování, sestavování a dalších činnostech nedochází pouze takříkajíc jednou v noci, ale průběžně. Právě proto je zde na místě slovo průběžné (angl. Continuous) [23].

3.3.4.1 CI – Continuous Integration (průběžná integrace)

Jednou z velkých výhod, které průběžná integrace nabízí, je fakt, že je vývojář o nefunkčnosti aplikace, kterou upravoval, informován takřka okamžitě, a to pomocí feedbacku, který může být zaslán například emailem. Tento email v sobě může obsahovat i další důležité informace, jako například podrobné informace o chybě, nebo o neúspěšnosti některých z testů. Tímto může na chybu zareagovat ve velmi krátkém čase, na rozdíl od situace, kdy by k nasazování a integraci docházelo pouze jednou za noc. Odpadá tak tedy i stres a nasazování v pozdních večerních hodinách. Podrobný postup průběžného nasazování, spolu se sestavením je uveden na následujícím obrázku.



Obrázek 7: Typické schéma nasazení CI, zdroj: [22]

Continuous Integration tvoří jeden ze základních pilířů při výstavbě dnešních komplexních aplikací [22]. Jejím hlavním přínosem je:

- Snížení rizika vnesení chyb
- Snížení redundantních manuálních procesů
- Zrychlení tempa vývoje
- Lepší transparentnost projektu

3.3.4.2 **CD – Continuous Delivery (průběžné dodání)**

Průběžné dodávání je termín popisující soubor technik, procesů a nástrojů, pomocí kterých lze dodávat software ve velmi krátkých vývojových cyklech s pozitivním efektem způsobujícím vyšší kvalitu a aktualizace softwarových aplikací [24].

Lze si to představit velmi jednoduše. U starších vývojových metod a praktik typu vodopád a další, probíhal vývoj aplikace tak, že musel projít celým vývojovým návrhem od analýzy, přes vývoj, testování a revize. Takto trval vývoj poměrně dlouhou dobu, než k zákazníkovi dorazila nová verze. S použitím průběžného dodání, lze tedy například každý den vydávat nové aktualizace s novými funkcemi a opravami stávajících chyb. Díky tomu lze stávající software poměrně pružně vyvíjet a zároveň s tím i odstraňovat chyby.

Průběžné dodávání se tedy snaží ověřit funkční i nefunkční požadavky na vyvíjený software, a to s velmi vysokým stupněm automatizace. Testy jsou aplikovány pokaždé a prováděny s každou změnou zdrojového kódu [24].

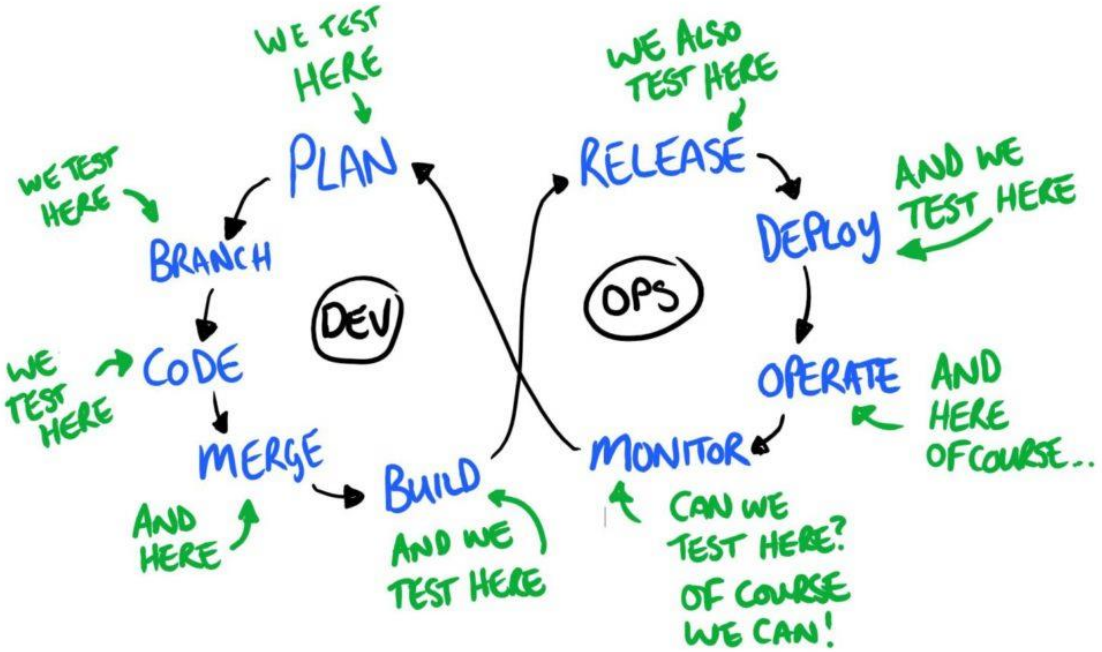
Vhodným kritériem, které zde lze sledovat bude tedy kolik minut na měsíc je přiděleno, pro běh těchto pipelines.

3.3.5 Automatické testování

Testování jako takové prošlo od modelů typu vodopád značným vývojem a posunulo se daleko blíže k vývojářům. Ve vodopádovém typu vývoje vytvářelo testování jeden bod v celém cyklu vývoje. Testování bylo zcela oddělené od vývojářského týmu a provádělo se obvykle na konci vývojového cyklu, a to před nasazením vyvíjené aplikace do produkčního prostředí [25].

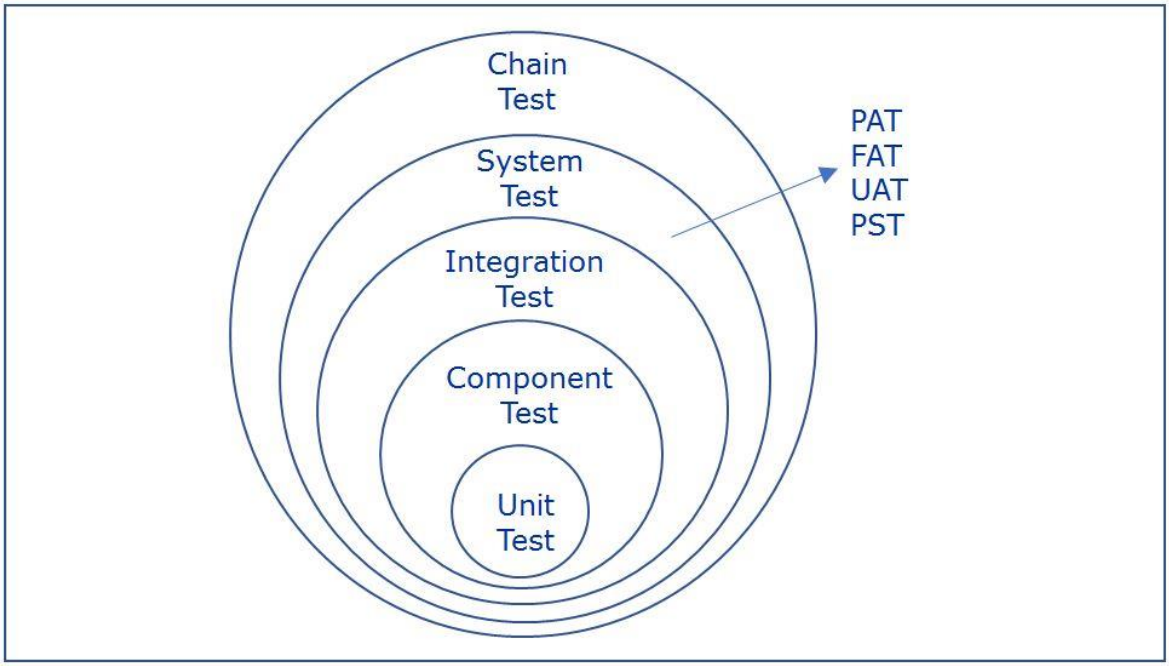
S nástupem agilního vývoje začal být software vyvíjen iterativně a inkrementálně, tedy již v této fázi se testovali jednotlivé přírůstky a zvýšila se tím efektivita vyvíjeného softwaru. Již zde se testování začalo považovat za část vývoje a testeři tak byly zařazeni blíže k vývojářům. Začali vznikat paralelní typy testování a tím se zvyšovala kvalita testovacích úkonů.

S příchodem DevOps se testy začali začleňovat stále dál a začalo se testovat už při samotném vývoji dané části softwaru. Testeři se v této fázi již nesoustředí pouze na integrační a verifikační testy. V DevOps modelu se začaly testy zaměřovat také na bezpečnost, výkonnost a další různé typy. Při pohledu na model DevOps začaly vyvstávat otázky kdy a co testovat. V tuto chvíli bylo možné testovat vše a všude, jak popisuje následující obrázek [25].



Obrázek 8: DevOps a testování, zdroj: [25]

Z předchozího obrázku je tedy zřejmé, že zde existuje velká škála možností pro to kde, co a kdo má testovat. Aby bylo možné se v těchto typech testů lépe orientovat, lze tedy využít možnosti tzv. osvědčených postupů (z angl. „best practises“). Jedním z těchto osvědčených postupů je tzv. rozdělení vrstev cibule, jenž je uveden na Obrázku 9.



Obrázek 9: The Onion layer of test types, zdroj: [31]

Tento obrázek vznikl syntézou osvědčených postupů a vzájemnými vztahy mezi vývojářskou pipeline a testovacím managementem [31]. Popisuje postup jednotlivých testů od jádra této cibule a vytváří tak obecné definice kategorií testů:

- Jednotkové testy – Testování jedné jednotky neboli jednoho drobného objektu, případu užití.
- Testy komponent – Z drobných objektů, nebo případů užití jsou skládány jednotlivé komponenty, které jsou testovány jako syntéza jednotlivých jednotek.
- Integrační testy – Tento typ testů kontroluje, zda spolu jednotlivé komponenty navzájem korektně kooperují.
- Systémové testy – Systémové testy kontrolují, zda byly naplněny požadavky na danou aplikaci jako na celek v podobě systému.
- Řetězové testy – Řetězovým testem je v poslední řadě zkontrolováno, zda spolu kooperují jednotlivé aplikace, tedy systémové celky mezi sebou.

Automatické i manuálně spouštěné testy se integrují buďto do jednotlivých pipeline, které byly zmíněné v předchozí kapitole, nebo tvoří součást zdrojového kódu vyvíjené aplikace. Nejlepším kritériem pro sledování vhodných prostředí, určených daným ekosystémem zde tedy bude sledováno, zda daný ekosystém umožňuje vykonávání těchto obecných kategorií jednotlivých testů, tedy zejména: jednotkové testy, testy komponent, integrační testy, systémové testy a řetězové testy.

3.3.6 Podpora a zpětná vazba

Zpětná vazba ze strany zákazníků je pro zvyšování efektivnosti a správnosti vyvíjených částí celé aplikace velmi důležitá, a proto hraje významnou roli v procesu DevOps. Vzhledem k rostoucím nárokům ze strany zákazníků a zvyšujícím se požadavkům na rychlost vývoje softwaru je zpětná vazba od zákazníka stěžejní [26].

Ekosystémy zastřešující celý proces DevOps, zmíněné v nadcházející kapitole, se zaměřují i na tuto specifickou část DevOps. Ku příkladu společnost Atlassian nabízí jako jeden ze systémů její rodiny tzv. Jira Service Desk, jenž je softwarem, který má na starosti právě podporu a zpětnou vazbu [7]. Jedním z kritérií pro vyhodnocení podpory a zpětné vazby tedy bude, zda daný ekosystém obsahuje v jeho celku nástroj pro podporu a zpětnou vazbu, tedy úkolový systém (z angl. „ticket systém“), který řeší přání a stížnosti ze strany

zákazníků. Dále také zda je tento systém zpoplatněn měsíčními, či jednorázovými poplatky, zda umí vytvářet reporty a notifikovat účastníky týmu pomocí emailu.

3.4 Vícekriteriální modely – metodika a identifikace kritérií

Vyhodnocení a výběr vhodné varianty z matice variant bude provedeno na základě využití modelů vícekriteriálního rozhodování, jelikož je zde nutné posoudit důsledky rozhodnutí na základě více kritérií. Aby bylo zaručeno maximálního objektivního přístupu a správnému výběru jedné, či více variant z množiny přípustných variant, bude zde využito modelu vícekriteriální analýzy variant.

V první řadě je zapotřebí zmínit, že existuje několik různých typů úloh, jak postupovat při získávání informací o jednotlivých variantách a jejich kritériích. Tyto úlohy lze dělit dle typu informace, získanou o preferencích mezi jednotlivými variantami. Situace, kdy žádná informace o preferencích neexistuje, je pro tento typ úkolu nepřípustná. Nominální informace dokáže rozlišit varianty na akceptovatelné a neakceptovatelné, avšak stále nedokáže vyhodnotit nejlepší variantu. Metody s ordinální informací umožňují získat přehled o pořadí jednotlivých variant a jejich kritérií, avšak neumožňují sdělit o kolik je daná varianta lepší než druhá a nelze podle ní tedy stanovit podrobnější rozhodnutí. Mezi tyto metody patří například metoda pořadí.

Posledním typem informací jsou informace kardinální, jejichž výstupem je vektor vah kritérií. Tyto metody a jejich typ informací mají kvalitativní i kvantitativní charakter. Pomocí nich lze tedy rozhodnout, která varianta a její kritéria jsou lepší než jiná a o kolik.

Mezi tyto typy informací se řadí metody Bodovací a Saatyho. Bodovací metoda určuje důležitost každého kritéria určitým počtem předem zvolené bodovací stupnice. Avšak tato metoda, vzhledem k jejímu subjektivnímu charakteru, je vhodná spíše v situacích, kdy je hodnocení jednotlivých kritérií prováděno více experty [6]. Naopak Saatyho metoda pro určení jednotlivých vah kritérií je vhodná právě tehdy, hodnotí-li je pouze jeden expert [6].

Vzhledem k výše zmíněným poznatkům bude využito pro stanovení vah kritérií a získání kardinální informace v podobě vektoru vah kritérií Saatyho metody.

Z tohoto vektoru lze získat nejlepší variantu a určit, o kolik je lepší než varianty ostatní. Saatyho metoda je metodou kvantitativního párového porovnání a je zde využita devíti bodová stupnice, kde:

1 – značí rovnocennost

3 – slabě preferované

5 – silně preferované

7 – velmi silně preferované

9 – absolutně preferované

Jednotlivá kritéria budou porovnána mezi sebou pomocí této stupnice a z výsledné matice se vypočítá míra konzistence. Dále se vypočítá normalizovaný geometrický průměr řádků a vypočtena suma vah jednotlivých kritérií. Následnou syntézou vah jednotlivých kritérií vznikne výsledná matice obsahující sumy vah jednotlivých variant. Tyto sumy v konečné řadě definují jednotlivé pořadí variant a hodnotu jejich preferencí.

Aby jednotlivé varianty splňovaly nároky obecných kritérií DevOps, budou vstupní data v první řadě očištěna o varianty, které těmto kritériím nevyhovují. Tím tedy dojde ke stanovení minimálních požadavků, které musejí být u jednotlivých kritérií variant naplněny, aby splňovali kvalitativní a kvantitativní podmínky obecných kritérií DevOps.

Syntézou metrik a obecných kritérií z předešlých kapitol došlo ke stanovení kritérií a jejich kategorizace:

- Obecná
    - Cena / uživatel / měsíc (CZK)
    - Jednorázová cena (CZK)
    - Maximální počet uživatelů
    - On-premise řešení
- Nástroje pro plánování a spolupráci
    - SCRUM
    - Kanban
    - Agile reporting
- Správa zdrojového kódu
    - Git repozitář
    - E-mailové notifikace
    - Velikost úložiště (GB)
    - Revize zdrojového kódu
    - Vkládání komentářů
    - Akceptace merge požadavků

- Sestavování (build)
    - Velikost úložiště artefaktů (GB)
    - Relase management
- CI/CD nástroje
    - CI/CD pipeline / minut / měsíc
- Automatické testování
    - Jednotkové testy
    - Testy komponent
    - Integrační testy
    - Systémové testy
    - Řetězové testy
- Podpora a zpětná vazba
    - Úkolový (ticket) systém
    - Reporty
    - E-mailové notifikace

4. Dostupná řešení a jejich vlastnosti

Pro praktickou realizaci výše popsaných moderních postupů softwarového vývoje je nutné využívat určitých nástrojů. V počátcích zavádění DevOps přístupu bylo nutné si sadu nástrojů a jejich propojení poskládat individuálně z většího množství malých, většinou open-source komponent. I když je tento přístup stále možný, představuje nezanedbatelné náklady zejména proto, že paleta existující nástrojů je velmi široká a pro zažínající tým velmi nepřehledná, viz Obrázek 2.

Protože každý problém zároveň přináší podnikatelskou příležitost, vznikly postupem času ucelené kolekce nástrojů s různým stupněm nezávislosti jednotlivých komponent, které integrují podporu vývoje stylem DevOps do dobře vyladěných řešení a jsou svými tvůrci nabízeny vývojářským týmům jako komerční služba. Nabízená řešení mnohdy navíc zapadají do ekosystému dalších vývojářských produktů a služeb jejich vydavatele.

Vzájemné porovnání těchto řešení je předmětem této práce. V této kapitole jsou stručně představeny jednotlivé ekosystémy, které byly vybrány jako vhodní kandidáti pro zamýšlené porovnání. Volba vychází zejména na základě agregace informací o tržním podílu jednotlivých nástrojů, které ekosystémy tvoří [32][33][34].

4.1 Microsoft ekosystém – Azure DevOps

Azure DevOps je cloudová platforma vytvořená společností Microsoft k účelům sestavování, testování, nasazování a správy služeb a aplikací napříč serverovými farmami této společnosti, které poskytuje jako IaaS (Infrastructure as a Service) pod obchodním označením Microsoft Azure.

Komponenty tvořící Azure DevOps, jednotlivě popsané níže, jsou poměrně ostře ohraničené a každá má svůj obchodní název. Současná podoba Azure DevOps vznikla postupný vývojem původních nástrojů Visual Studio Team Services (VSTS) a Team Foundation Server (TFS), které na jednu stranu disponovaly dobrou integrací s ostatními produkty společnosti Microsoft, na druhou stranu však trpěly problémem velké uzavřenosti vůči konkurenčním programovacím jazykům a nástrojům. Přerod těchto nástrojů v Azure DevOps, který zachovává zmíněnou vlastnost první a eliminuje vlastnost druhou, souvisí zejména s nedávnou radikální změnou politiky společnosti Microsoft, kdy konkurenční

technologie přestávají být nenáviděny, ale naopak synergicky přijímány, viz nové moto „Microsoft loves Linux“ [35].

Díky tomuto posunu nyní Microsoft Azure podporuje různé programovací jazyky, frameworky a nástroje třetích stran [27], přičemž si zachovává excelentní integraci s ostatními produkty Microsoftu. Proto je vhodnou volbou zejména pro týmy, jejichž IT infrastruktura je na těchto produktech postavena. Ačkoliv je Azure DevOps a vlastně i celé Microsoft Azure nabízeno primárně jako cloudové řešení, je možné přejít i na on-premise variantu a hostovat tyto služby samostatně na vlastních zařízeních. I tato varianta, avšak nadále zůstává spjatá s určitou částí zdrojů poskytovaných výhradně přes online služby Microsoft Azure.

4.1.1 Nástroje pro plánování a spolupráci

Služby pro plánování a spolupráci realizuje v rámci Azure DevOps komponenta s obchodním názvem Azure Boards. Disponuje všemi funkcemi, které lze u nástroje tohoto typu očekávat, přičemž podporuje čtyři typy vývojového procesu:

- Základní (kanban board s úkoly bez formální metodiky),
- Agilní (podpora backlogu, sledování dle user stories, odhady),
- Scrum (stejné jako agilní s podporou sledování metrik specifických pro Scrum)
- CMMI (velmi formální vývoj stejnojmennou metodikou, obdoba ISO norem)

Projekt lze pojmout jako veřejný nebo uzavřený v rámci organizace. Řízení oprávnění pro všechny úkony v rámci plánovací agendy může být velmi detailní a lze do něj zahrnut i složité vnitropodnikové směrnice, což je zajímavé zejména pro korporace, jejichž potřebám Microsoft tradičně velmi dobře rozumí. Systém oprávnění však může být matoucí pro malé začínající týmy, pro které bude granularita oprávnění jistě značně přehnaná. I tyto týmy však mohou těžit z velmi propracovaného systému oznámení, které Azure Boards má. Zajímavostí je možnost zřízení účtu tzv. stakeholdera, což je omezený účet pro zástupce klienta, jenž mu umožní sledovat postup prací a zúčastnit se připomínkování vývoje, což ocení i malí vývojáři pracující v duchu agilních postupů se silnou účastí jejich klienta.

Azure Boards je dle očekávání dobře integrována s ostatními součástmi Azure DevOps. Funkce jako např. vytváření větví v repozitáři pro nové úkoly nebo naopak

uzavírání úkolů při sloučení (angl. merge) větví repozitáře jsou proto samozřejmostí. Mimo to disponuje Azure Boards dobrou integrací s GitHubem a nabízí též vlastní REST API, a tedy i téměř neomezené možnosti integrace s vlastními řešeními umožňující například automatizaci zadávání bugů. Funkce Azure Boards je dále možné rozšiřovat instalací plug-inů přes interní Marketplace, např. pro integraci s externími nástroji pro billing a timetracking, které Azure Boards sama o sobě nenabízí, což může být pro menší týmy realizující více menších zakázek nepříjemný nedostatek.

Z obchodního pohledu nemá Azure Boards jiná omezení, než je počet uživatelů, který se odvíjí od počtu uživatelů zvoleného Azure DevOps plánu.

4.1.2 Správa zdrojového kódu

Služby pro správu zdrojového kódu zajišťuje komponenta s obchodním názvem Azure Repos. Podporováno je verzování pomocí Git, jakožto všeobecně používaného standardu, a Team Foundation Version Control (TFVC), což je standard pro centralizovanou správu zdrojového kódu specifický pro vývojové prostředí Microsoft Visual Studio. Naopak nejsou podporovány standardy Mercurial nebo SVN.

Azure Respos všeobecné koncepty systému Git rozšiřuje o integraci s ostatními součástmi Azure DevOps, o propracované řízení oprávnění k repozitáři a jednotlivým větvím (tzv. branch policies), které lze efektivně použít pro vynucení politiky správy zdrojového kódu používané týmem (např. git flow), a o kolaborativní nástroje – komentáře, code-review, schvalování a řešení konfliktů.

Azure Boards disponuje možnostmi integrace s populárními vývojovými prostředími (tzv. IDE), zejména pak s VisualStudio, InteliJ, XCode, Eclips, Atom a Sublime. Rovněž je možní využít API a tzv. web hooks k automatizaci činností týkající se správy zdrojového kódu.

Obdobně jako Azure Boards, nemá ani Azure Repos v zásadě jiná omezení, než je počet uživatelů zvoleného Azure DevOps plánu a neomezuje počet ani velikost privátních Git repozitářů, samostatně ji však nelze koupit.

4.1.3 Nástroje pro sestavování (build)

Sestavování, které v rámci rozdělení pro potřeby této práce zahrnuje také správu balíčků a nasazování, realizují v rámci Azure DevOps dvě komponenty: Azure Pipelines a Azure Artifacts.

Azure Pipelines je především robustním nástrojem pro realizaci CI/CD, proto je podrobněji popsána v příslušné sekci dále v textu. Z pohledu buildu a nasazení je důležitá zejména informace o podpoře technologií. Azure Pipelines umí zajistit sestavení zdrojů těchto technologií: .NET Core, ASP.NET, Anaconda, Android, C/C++ (GCC a VC++) Docker kontejnerů, Go, Java, Javascript a Node.js, PHP, Python, Ruby, UWP, Xamarin a Xcode.

Cílem pro nasazení mohou být: Kubernetes, Azure Stack, Azure SQL database, Azure Web App, Web App for Containers, npm, NuGet, VMware, Windows VM, Linux VM i obecné virtuální stroje. Je tedy podporováno nasazování jak do cloudu, tak na on-premise infrastrukturu. Každá s uvedených technologií má specifická nastavení a jejich popis zdaleka přesahuje rámec této práce. Obecně však lze vyzvednout propracovaný mechanismus Azure Pipelines, který je velmi flexibilní a umožňuje formalizovat i velmi složitou politiku schvalování a kontrol, které zajistí ochranu před nežádoucím nasazením artefaktů s nedostatečnou kvalitou (které např. neprošli úspěšně testy).

Azure Artifacts oproti tomu představuje především úložiště balíčků pro jejich sdílení a distribuci v rámci organizace. Podporuje balíčkovací systémy NuGet, npm, Maven a Python (PIP). Kromě toho je možné vytvářet i tzv. univerzální balíčky, které představují kolekci několika souborů, která má určité jméno a verzi. Díky tomu je možné Azure Artifacts používat jako hosting balíčků pro prakticky jakýkoliv druh dat.

Balíčky jsou organizovány do tzv. feedů, oprávnění k nim lze detailně spravovat. To umožňuje vytvářet jak feedy veřejné, tak uzavřené pro interní vývoj. Balíčkovací systémy inherentně musí zajistit tzv. imutabilitu (neměnnost) obsahu balíčku určité verze, což bezchybně Azure Artifacts zajišťuje a nadto nabízí funkci „odpadkového koše“, kdy v případě smazání či unlistingu určitého balíčku drží jeho kopii ještě po dobu 30 dnů pro možnost obnovy.

Velmi zajímavou funkcí je tzv. upstreaming: Ten umožňuje použít privátní feed jako mezičlánek pro přístup k feedům veřejným, které jsou obvykle zdrojem open-source knihoven. Privátní feed se díky tomu začne chovat jako cache pro veřejné balíčky používané našimi projekty, což může zachránit vývoj v případech, kdy veřejný feed přestane fungovat, nebo je v horším případě tzv. otráven škodlivým kódem (což je problém, který se v posledních letech objevuje stále častěji [36]).

Z obchodního pohledu je u Azure Artifacts cena přímo úměrná potřebě úložného prostoru a jde o službu, kterou je možné zakoupit i samostatně. Základní verze zdarma,

která je i součástí Azure DevOps plánů zahrnuje 2 GB úložného prostoru pro balíčky s tím, že rozšíření je možné dokoupit za ceny odstupňované dle velikosti prostoru, viz [37].

4.1.4 CI / CD nástroje

V rámci ekosystému Azure DevOps zajišťuje úkoly CI/CD komponenta s obchodním názvem Azure Pipelines. Jde o univerzální cloudovou službu, která umožňuje provádět automatizované sestavování a testování prakticky jakýchkoliv projektů používajících mnoho jazyků. Hlavními podporovanými jazyky jsou Python, Java, JavaScript, PHP, Ruby, C#, C++ a Go. Podporovány jsou platformy Windows, Linux a Mac.

Azure Pipelines skládají DevOps procesy z jednotlivých kroků (angl. step), které jsou součástí úloh (angl. job). Úlohy jsou organizovány do etap (angl. stage), jejichž vzájemná posloupnost tvoři pipeline. Běh pipeline (angl. run) může být spouštěn ručně, častěji však automaticky na základě nějaké události (angl. trigger).

Úkolů, které je možné pomocí Azure Pipelines provádět je celá řada – od automatizovaného testování až po automatizované nasazení, jak bylo popsáno v předchozím bodě výše. Jde o nástroj s vysokou flexibilitou, který umožňuje formalizovat i složité procesy schvalování a kontrol. Konfigurace se zapisuje především s pomocí jazyka YAML, k dispozici je však také příjemné grafické uživatelské rozhraní. K dispozici jsou též pokročilé možnosti integrace s dalšími systémy speciálně zaměřené na vývoj opensource projektů, zejména pak s GitHubem, které však nejsou předmětem zájmu této práce. Logicky očekávatelné je rovněž dobré provázání s ostatními produkty společnosti Microsoft, zejména pak s Microsoft Azure.

Samotný běh, tedy v konečném důsledku vykonání posloupnosti kroků, probíhá na tzv. agentech – výkonových jednotkách, které jsou nasazeny v určitém prostředí. Azure Pipelines nabízí agenty cloudové, jejichž čas je možné zakoupit, nicméně podporuje také agenty lokální, které běží na on-premise infrastruktuře. Protože Azure Pipelines v zásadě neklade omezení z hlediska používaných technologií, je z obchodního pohledu důležité právě to, kde k běhu agentů dochází a jaká je kapacita pro paralelně prováděné úlohy.

U cloudové varianty je v základu zdarma 1 800 minut jedné paralelní úlohy, každá další paralelní úloha je za 40 USD bez omezení běhových minut. U on-premise varianty je v základu zdarma jedna paralelní úloha bez omezení běhových minut (což je logické vzhledem k tomu že je využívána pouze infrastruktura klienta), každá další paralelní úloha

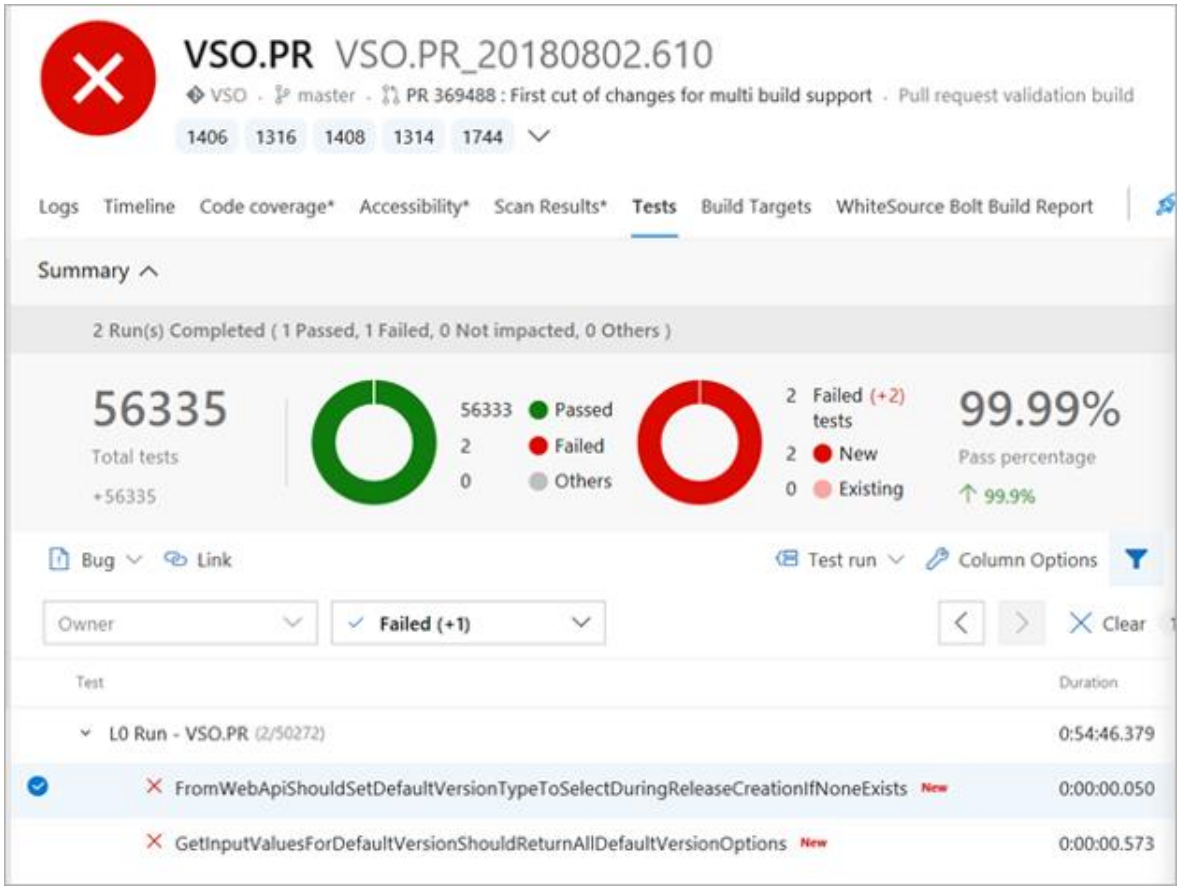
za 15 USD bez omezení běhových minut [37]. Komponentu Azure Pipelines je možné zakoupit samostatně nebo v Azure DevOps plánu (bez rozdílu podmínek).

4.1.5 Automatické testování

Testování je v rámci Azure DevOps zajištěno dvěma komponentami – automatizované testy pomocí Azure Pipelines a manuální a akceptační testy pomocí rozšiřující komponenty Azure TestPlans.

Realizace automatizovaných testů závisí především na použitých nástrojích, jejichž nasazení se odvíjí od technologie toho, kterého projektu a jeho architektury. Díky univerzálním vlastnostem Azure Pipelines je možné automatizovaně provádět širokou škálu testů, zejména pak jednotkové testy a testy komponent.

Za tímto účelem je přítomna podpora zejména pro výstupy testovacích nástrojů CTest, JUnit, PHPUnit, NUnit 2 a 3, Visual Studio Test (TRX), and xUnit 2. S pomocí Azure Pipelines je možné realizovat i automatizované integrační, systémové i řetězové testy, např. také UI testy s pomocí frameworku Selenium, nicméně jde již o poměrně pokročilou úlohu. Velkou výhodou je pak propracované zpracování výstupů testování s velmi přehlednými statistikami, viz Obrázek 10.



Obrázek 10: View tests tab, zdroj: [38]

Azure DevOps nabízí také zajímavou rozšiřující komponentu Azure TestPlans pro formální organizaci manuálních testů s lidským faktorem, zejména pak akceptačních testů.

Testování tohoto druhu spadá do kategorie systémových a řetězových testů, kdy se testuje celý produkt přes jeho rozhraní vystavené směrem ke koncovému uživateli. Tato komponenta umožnuje realizaci plánovaných manuálních testů, uživatelských akceptačních testů tzv. exploratory testů pro product owners v rámci teamu a zejména stakeholder feedback testů pro prezentaci funkcionality samotným koncovým zákazníkem.

K dispozici je kompletní management pro přípravu testových plánů a rozšíření do webových prohlížečů umožňující provádět videozáznam, snímky obrazovky a komunikaci mezi testery a vývojářským teamem. Provedení manuálních testů lze také naplánovat automaticky v rámci Azuire Pipelines. Objevené chyby je možné nechat automaticky zanést do Azure Boards jako bug a naplánovat jejich opravu.

Obchodní hledisko komponenty Azure Pipelines je podrobně popsáno v předchozím bodě výše. Komponenta Azure TestPlans je dostupná pouze v rámci rozšířeného plánu Azure DevOps.

4.1.6 Podpora a zpětná vazba

Ekosystém Azure DevOps nenabízí ucelenou podporu pro zpracování zpětné vazby od koncových uživatelů produktu. Není k dispozici žádný tiketový systém, ani podobná funkce. Jsou však k dispozici rozšíření pro integraci nástrojů podpory od třetích stran jako je např. Zendesk, FreshService nebo AutoTask, a s pomocí API je možné do Azure Boards integrovat také spolupráci s dalšími systémy.

4.2 Atlassian ekosystém

Atlassian je softwarová společnost zabývající se tvorbou nástrojů pro spolupráci, která vybudovala zajímavé portfolio vzájemně spolupracujících služeb nejen pro softwarové vývojáře, ale také pro IT podporu, HR, Marketing, právní oddělení a obecně pro jakýkoliv team, který nějakým způsobem potřebuje organizovat svoji práci. Základním produktem Atlassianu je project management aplikace Jira, která je jedním z nejpopulárnějších nástrojů pro řízení projektů obecně (spolu s odlehčenou verzí Trello) [32].

Ekosystém Atlassianu prošel určitým vývojem, v rámci něhož docházelo v nedávné době k postupnému slaďování vzhledu a hlubší integraci jednotlivých komponent, což je dáno tím, že portfolio Atlassianu bylo postupem let sestaveno především postupnými akvizicemi jiných produktů. Jednotlivé součásti Atlassian ekosystému jsou proto stále samostatnými produkty a je možné z nich sestavit skládačku přesně podle představ konkrétního teamu.

Vzhledem tomu, že Atlassian není historicky spojen s výrobci operačních systémů nebo cloudovými poskytovateli, nenese si břemeno podpory legacy systémů, jako je tomu např. u produktů společnosti Microsoft. Z obchodního pohledu je ekosystém Atlassianu zajímavý také poměrně vstřícnou cenovou politikou pro malé týmy.

4.2.1 Nástroje pro plánování a spolupráci

Ke správě procesů při vývoji software slouží vlajková loď Atlassianu – nástroj Jira Software. Jira Software nabízí veškerou podporu pro vývoj software s pomocí agilních metodik. Samozřejmostí je podpora metodiky Scrum, vč. Scrum boardu, kanban boardu, roadmaps. K dispozici je též propracovaný agile reporting – burndown/burnup chart, sprint report, cumulative flow diagram, velocity chart. Kromě očekávatelných funkcí potěší podpora pro definici vlastních worflow, které je možné snadno přizpůsobit individuální

politice teamu. Jira Software umožňuje napojení na repozitáře spravované prostřednictvím služby Bitbucket (Atlassian) a GitHub (externí poskytovatel). Směrem na opačnou stranu jsou k dispozici integrační rozšíření pro populární IDE. Další integrace je možné realizovat buď pomocí pluginů dostupných přes Marketplace. Zajímavostí je vlastní dotazovací jazyk JQL (Jira Query Langue) s jehož pomocí je možné integrovat Jira Software prakticky s jakoukoliv stávající infrastrukturou.

Stejně jako všechny produkty Atlassianu, klade Jira Software ve svém rozhraní důraz na jednoduchost a nabízí přehledné uživatelské rozhraní. Je vhodné doplnit, že podobné funkce může nabídnout nástroj Trello, který je ovšem zevšeobecněnou a značně osekanou verzí kanban boardu a pro účely této práce jej proto pomíjíme.

Jira Software může být provozován jak v předplaceném cloudu Atlassianu, tak samostatně jako on-premise řešení. Dostupná je také verze Data Center pro hostování v cloudu Amazon AWS a MS Azure. Z obchodního hlediska je důležitá volba způsobu provozování.

U cloudové verze je k dispozici varianta zdarma (do 10 uživatelů a 2 GB prostoru), ve které však schází jakákoliv podpora, řízení uživatelských práv, a především možnost přístupu pro osoby mimo organizaci, což jsou typicky zákazníci vývojářské firmy. Cloudová verze je dále dostupná ve variantách Standard (250 GB a standardní podpora) a Premium (neomezený prostor, dostupnost 99,9 %, nonstop podpora), jejichž cena je nelineárně odstupňovaná podle počtu uživatelů a frekvence placení [39] s nejlevnější cenou 100 USD pro 1-10 uživatelů verze Standard při roční platbě.

U on-premise verze jsou dostupné varianty Server (trvalá licence s aktualizacemi po dobu jednoho roku) a Data Center (roční licence vč. aktualizací, napojení na vnitropodnikový SAML, optimalizace pro AWS a Azure nasazení a prémiová podpora).

Ceny jsou odstupňované dle počtu uživatelů [39]. Varianta Server začíná na 10 USD pro až 10 uživatelů, další stupeň pro až 25 uživatelů za 3.500 USD. Jde přitom o jednorázovou platbu za trvalou licenci, což je v oblasti těchto nástrojů poměrně unikátní nabídka, která může představovat zajímavou volbu pro malé začínající společnosti. Varianta Data Center začíná na 500 uživatelích s cenou 20.400 USD za rok, což odpovídá jejímu zaměření na střední a velké podniky.

4.2.2 Správa zdrojového kódu

Správu zdrojového kódu obstarává v Atlassian ekosystému nástroj s obchodním názvem Bitbucket. Svými funkcemi pokryje všechny základní potřeby očekávané u nástroje této kategorie. Umožňuje správu neomezeného počtu soukromých repozitářů typu Git – jiné typy však nepodporuje (kromě Mercurial, ale pouze u cloudové verze). Nadto umožňuje uspořádání repozitářů do projektů, podporu pro řízení oprávnění jednotlivých větví, kontrolu merge procesů, zobrazování rozdílů (tzv. diff), prohledávání kódu a kolaborativní anotaci jednotlivých položek nebo i jednotlivých řádků kódu. Zajímavostí je integrovaná podpora Git Large File Storage (LFS) a verzované úložiště velkých soborů, díky kterému je možné paralelně s kódem spravovat také např. zdrojové soubory grafiky a výstupy programů jako je Adobe Photoshop či Illustrator.

Přítomnost dalších funkcí se však dramaticky liší v závislosti na zvolené verzi dle způsobu hostování – cloudová verze obsahuje jako integrální součást nástroje pro build a CI/CD, základní issue tracking a tvorbu interní wiki, které je u on-premise verze nutné řešit zakoupením samostatných nástrojů s obchodními názvy Bamboo, Jira Software, a Confluence. Tyto funkce nástroje Bitbucket jsou podrobněji popsány v příslušných sekcích dále v textu. Podrobný přehled funkcí v jednotlivých verzích viz [40].

Cloudová verze je navíc dostupná ve dvou variantách Standard a Premium, která navíc obsahuje zejména podporu pro vynucení kontroly merge procesů a řízení oprávnění k deployment operacím.

Bitbucket je možné provozovat jak v předplacené cloudové verzi, tak v on-premise verzi. Cloudová verze má limit na velikost jednoho repozitáře 2 GB (LFS se nezapočítává). Obchodní model je obdobný jako u nástroje Jira Software: Cloudová verze je k dispozici ve variantách zdarma (až 5 uživatelů, 1 GB LFS, 50 CI/CD minut měsíčně), Standard (3 USD za uživatele a měsíc, 5 GB LFS, 2.500 CI/CD minut měsíčně) a Premium (6 USD za uživatele a měsíc, 10 GB LFS, 3.500 CI/CD minut měsíčně, rozšířené funkce pro řízení oprávnění a monitoring). Placené varianty mají od 100 uživatelů výše odstupňované zvýhodněné ceny [41].

On-premise verze je podobně jako u Jira Software dostupná ve variantách Server (trvalá licence s aktualizacemi po dobu jednoho roku) a Data Center (roční licence vč. aktualizací, napojení na vnitropodnikový SAML, smart monitoring a prémiová podpora). Cenově začíná varianta Server na 10 USD jednorázově pro až 10 uživatelů, varianta Data

Center na 1.980 USD za rok pro až 25 uživatelů. Ceny pro větší počty uživatelů jsou dále odstupňované [41].

4.2.3 Nástroje pro sestavování (build), CI / CD nástroje a automatické testování

V ekosystému Atlassian jsou úkoly sestavování, deploymentu, testování a řešení CI/CD realizovány integrálně pomocí jednoho nástroje, proto je nyní budeme diskutovat společně v rámci jedné sekce. Tyto úkoly v závislosti na zvolené variantě hostování řeší buď nástroj Bitbucket pro cloudovou variantu nebo dedikovaný nástroj Bamboo pro on-premise variantu, přičemž obě varianty nejsou vzájemně zaměnitelné zejména z hlediska podporovaných technologií. Tato dvoukolejnost je dána pravděpodobně historií portfolia ekosystému Atlassian, které, jak bylo zmíněno, vzniklo zejména potupnou akvizicí různých produktů.

4.2.3.1 Řešení pomocí Bitbucket Pipelines

V rámci nástroje Bitbucket v cloudové verzi je k dispozici služba Bitbucket Pipelines, která umožňuje provádět výše uvedené úkoly. Jejím základem je framework pro vytváření a správu pipeline, které jsou složeny z jednotlivých kroků (angl. step), jejichž činnost je definována skriptem. Tento skript přitom operuje zejména s repozitářem samotným – při průchodu pipeline spustí každý krok vlastní Docker kontejner s klonem repozitáře, nad kterým se spustí příslušný skript.

Skripty samotné jsou přitom psány přímo ve zvoleném programovacím jazyce, což je velice příjemné pro vývojáře, protože odpadá nutnost učit se proprietární skriptování. Tento přístup na jednu stranu poskytuje vysokou flexibilitu, na druhou stranu zde není k dispozici přívětivé uživatelské rozhraní pro definici činnosti kroků. K dispozici jsou jazyky a nástroje: PHP (Laravel), JavaScript/Node.js, Java (Maven/Gradle), Python, Ruby, C#, C++ (make), Clojure, Docker, Go, Haskel, .NET Core, Rust, Scala, Perl a Erlang s tím, že je možné použít i další jazyky které mohou být sestaveny na Linuxu pomocí Docker image. Spouštění jednotlivých pipeline je navázáno na události ve správě zdrojového kódu nebo v systému řízení projektu Jira Software. Konfigurace pipeline je definována v jazyce YAML.

Bitbucket Pipelines dále podporují automatické sestavování artefaktů s nasazováním na tyto cíle: Bitbucket, Amazon AWS, Ansible Tower, Microsoft Azure,

Docker, Firebase, FTP, Google Cloud, Heroku, Kubernetes, Maven, npm, Read the Docs a SCP.

Automatizované testování je možné v rámci skriptů zajistit pomocí testovací frameworků určených pro konkrétní programovací jazyk s tím, že Bitbucket Pipelines umí zpracovat jakýkoliv výstup v XML xUnit-compatible formátu (s výjimkou xUnit.net). Explicitně jsou podporovány xUnit, JUnit, Maven Surfire a Failsafe, Gradle, PHPUnit, Moca pro Node.js a Unittest pro Python. Všechny tyto nástroje jsou zaměřeny na provádění jednotkových a komponentových testů, nicméně díky zajímavé možnosti napojení na databázi pomocí pipeline s nimi lze realizovat i testy vyšších vrstev, byť s určitými obtížemi. Zcela ovšem chybí podpora pro akceptační testy či manuální testovací scénáře.

Zásadním omezením Bitbucket Pipelines je absence podpory pro technologie rodiny .NET Framework, což je problém pro teamy pracující s příbuznými platformami (.NET Core však lze s určitými omezeními použít). Rovněž není možné sestavování aplikací pro Windows, macOS, nebo iOS.

Bitbucket Pipelines je součástí nástroje Bitbucket, který byl z obchodního hlediska popsán výše. V této souvislosti je však vhodné doplnit cenu za dokoupení běhového času, která činí 10 USD za měsíc a 1.000 minut.

4.2.3.2 **Řešení pomocí Bamboo**

Atlassian Bamboo je samostatný serverový on-premise nástroj, který realizuje CI/CD služby včetně sestavování, deploymentu a testování na vlastních zařízeních vývojářského subjektu. Na rozdíl od řešení Bitbucket Pipelines jde o plnohodnotný a robustní CI/CD nástroj, který nabízí všechny funkce očekávatelné u tohoto typu produktu. Bamboo podporuje nasazení na platformách Windows, Linux i Mac OS, což v důsledku umožňuje podporu prakticky jakéhokoliv programovacího jazyka či nástroje.

Bamboo používá model server-agent, kdy na jednom stroji běží centrální řídící serverová aplikace, která orchestruje jednoho nebo více agentů, na kterých probíhá běh samotných úloh. Workflow CI/CD je organizováno v několika úrovních – projekt, plán, stage, job a task. Neuralgickým bodem je stage, kde dochází jednak k paralelismu běhu, jednak k sestavování artefaktů, které poté využívají následné stage. Výkonným bodem jsou jednotlivé tasky, které mohou být různého typu a provádět sestavení, tetování a jeho vyhodnocení, deployment, nastavování proměnných, iniciovat operace SCM nebo provádět

speciální úlohy připravené programátorem. Podporováno je sestavení pomocí Ant, Grails, Maven, MSBuild, NAnt, Visual Studio, Xcode a Fastlane, příp. také pomocí shellových skriptů (CMD, Bash) či vlastních spustitelných nástrojů. Nástroj Bamboo je založen na platformě Java a je pro ni i optimalizován. Tento fakt je také určující pro skutečnost, že konfigurace pipelines je řešena pomocí skriptů v jazyce podporovaném JVM – Java, Groovy, Scala nebo Kotlin, formát YAML je možné použít až od verze 6.3 výše.
K dispozici jsou nástroje pro reporting a statistickou analýzu všech automatizovaných procesů.

V otázce deploymentu je poskytováno univerzální rozhraní, které neomezuje v použití libovolného deployovacího nástroje, který je obvykle k dispozici jako součást cílového prostředí. K dispozici jsou přednastavené vzory tasků pro Tomcat, Maven a Ant, dále pak pro SCP, SSH a shell skripty. K dispozici je též speciální handler task pro manipulaci s artefakty a jejich ukládání mj. na Amazon S3.

Automatizované testování vč. jeho vyhodnocení je možné provádět pomocí JUnit, MBUnit, MSTest, NUnit, PHPUnit, TestNG, OCUnit, RSpec a CPPUnit. Pro další testovací nástroje jsou k dispozici komunitní rozšíření, příp. je možné alternativní testovací nástroj nakonfigurovat pro výstup ve formátu podporovaným alespoň jedním uvedeným modulem. Vzhledem k velké variabilitě tasků je možné provádět v podstatě jakékoliv kategorie testů.

Z obchodního pohledu je Bamboo možné zakoupit pouze v on-premise variantě. Základní verze s cenou 10 USD jednorázově je omezena na 10 úloh (jobs) a pouze místní agenty (jichž však může být neomezený počet), další varianty odstupňované dle počtu vzdálených agentů již nejsou omezeny na počet úloh. Nejlevnější pro jednoho vzdáleného agenta vyjde na 1.270 USD [42].

4.2.4 Podpora a zpětná vazba

Vzhledem k tomu, že se produkty Atlassianu nezaměřují pouze na vývoj software je pro zajištění podpory a zpětné vazby možné využít řešení s obchodním názvem Jira Service Desk. Jde o samostatnou službu, která je však dobře provázána s nástrojem pro správu projektu Jira Software, jak již napovídá její název.

Jira Service Desk je nástroj primárně určený pro poskytování IT podpory i mimo oblast vývoje software. Součástí jsou tak kromě samotného jádra ticketovacího systému také funkce pro zachytávání a reporting incidentů, tvorbu workflow pro operátory (v rámci

tohoto nástroje jsou nazýváni „agenti“), komunikaci po mnoha kanálech (kromě e-mailu také prostřednictvím sociálních sítí, SMS nebo pomocí živého chatu), funkce zákaznické samoobsluhy, hodnocení spokojenosti s poskytnutou podporou a mnoho dalších [43].

Z pohledu tématu této práce je důležitá dobrá provázanost na organizaci práce směrem k vývojářskému týmu, kdy mohou být hlášené problémy podkladem pro založení úkolů a bugreportu v systému pro řízení projektu. Jira Service Desk dále disponuje pokročilým reportingem součástí kterého je i možnost kontrolovat dodržování plnění SLA vůči jednotlivým klientům a na základě něj zpracovávané podněty prioritizovat. Dostupná funkcionalita nástroje Jira Service Desk tudíž nejen splňuje všechny sledované aspekty, ale naopak je značně převyšuje. Zajímavou možností je propojení s dalším nástrojem Confluence, který slouží pro tvorbu interní wiki, kterou mohou využívat jak vývojáři interně v rámci teamu, tak koncoví zákazníci jako přívětivou samoobslužnou bázi znalostí.

Nástroj Confluence je však samostatně placenou službou, a protože jeho funkce již přesahují rámec zkoumaný touto prací, nebude jeho zakoupení uvažováno.

Z obchodního pohledu je nástroj Jira Service Desk dostupný k zakoupení ve verzi pro cloud a on-premise, obchodní model je podobný nástroji Jira Software.

Cloudová verze je dostupná ve variantách Free, Standard a Premium, které se liší přítomností auditovacích protokolů (ne/ne/ano), garancí dostupnosti 99,9 % (ne/ne/ano), velikostí úložiště (2 GB/250 GB/neomezeně), možnostmi podpory (komunitní/standardní/nonstop) a samozřejmě také měsíční cenou (zdarma / 20 USD za agenta / 40 USD za agenta). K dispozici jsou množstevní slevy v nelineárně odstupňovaných pásmech dle počtu agentů, zvýhodněná je rovněž roční frekvence placení [43].

On-premise verze dostupná ve variantách Server (trvalá licence s aktualizacemi po dobu jednoho roku) a Data Center (roční licence vč. aktualizací, clustering a podpora při řešení havárií). Cenově začíná varianta Server na 10 USD jednorázově pro až 3 agenty, varianta Data Center na 13.200 USD za rok pro až 50 agentů. Ceny pro větší počty agentů jsou dále odstupňované [44]. Z hlediska možné provázanosti nástrojů je vhodné zmínit, že počet agentů nemusí odpovídat počtu uživatelů jiných produktů – uživatelé Jira Software bez agentské licence mohou vykonávat omezené činnosti, zejména komunikovat s agenty při řešení požadavků. Naopak pokud bude jedna osoba plnohodnotně pracovat zároveň na vývoji i na podpoře, bude muset mít licenci pro každý využívaný produkt.

4.3 GitLab

Nástroj GitLab (neplést s konkurenční službou GitHub) je webový Git repozitář, který v průběhu času postupně obrostl mnoha funkcemi a nyní představuje ucelené a komplexní řešení zaměřené speciálně na přístup DevOps s ambicí postihnout jeho celý životní cyklus. Na rozdíl od výše popisovaných nástrojů je GitLab specifický v tom, že netvoří ekosystém nástrojů, ale je sám jedním koherentním nástrojem pro všechny úkoly související s vývojem software a je tak i prezentován. Této vlastnosti mimo jiné vděčí za svůj dominantní obchodní podíl v on-premise řešeních [45].

Vzhledem k tomu, že GitLab je nabízen jako ucelený produkt, je nutné zhodnotit obchodní hledisko globálně pro všechny sledované podskupiny funkcí. GitLab je nabízen jak v cloudové verzi, tak v on-premise verzi. Cenově ani funkčně mezi verzemi není rozdíl, kromě přítomnosti specifických podpůrných funkcí u on-premise verze, jako je například možnost napojení na LDAP.

On-premise verze logicky též nemá žádná omezení týkající se prostoru ani času běhu CI. Obě verze jsou k dispozici ve čtyřech variantách dle funkční úrovně. Vzhledem k velkému množství funkcí [46] jsou pro přehlednost uvedeny pouze zásadní rozdíly relevantní ke zkoumané funkcionalitě:

- Free (Core) – zdarma, 2.000 minut CI, neobsahuje pokročilé funkce řízená projektu a správy zdrojového kódu
- Bronze (Starter) – 4 USD, 2.000 minut CI, neobsahuje nástroje podpory a zpětné vazby
- Silver (Premium) – 19 USD, 10 000 minut CI, neobsahuje roadmapy a pokročilý reporting
- Gold (Ultimate) – 99 USD, 50 000 minut CI

  Veškeré ceny jsou počítány za uživatele a měsíc a jsou účtovány ročně.

4.3.1 Nástroje pro plánování a spolupráci

Plánování je v rámci GitLabu zaměřeno výhradně na agilní metodiky a nabízí jejich širokou podporu. Řešení není explicitně zaměřeno na konkrétní metodiky, ale poskytuje velkou flexibilitu pro realizaci prakticky jakékoliv agilní metodologie, kterou může tým používat.

Přítomny jsou tak všechny základní funkce pro realizaci kanban boardu a Scrumu (issues, tasks) s rozšířením o organizaci do vyšších funkčních celků (epics a milestones) nebo podle štítků (labels). Sprinty metodiky Scrum je možné realizovat v rámci plánů nazývaných roadmaps. K dispozici jsou též nástroje pro agilní bodový scoring, estimaci úkolů a reporting (zejména oblíbený burndown chart).

Ke všem prvkům je přitom možné vést v diskuse ve vláknech, k dispozici jsou nástroje pro její moderování. Vysoký stupeň integrace umožňuje velmi detailní trasování práce v návaznosti na SCM a CI/CD. Oproti konkurenčním řešením je již v základu nabízeno řešení interní wiki a také podpora tzv. Enterprise Agile Frameworks pro použití agilních metodik v řízení ostatních firemních procesů, které přesahují samotný vývoj. Díky tomu je plánování řešené pomocí GitLabu velmi dobře škálovatelné v průběhu času a dobře připravené na postupný růst týmu nebo změnu orientace na jinou metodologii [47].

4.3.2 Správa zdrojového kódu

Protože je celý GitLab postaven okolo Git repozitáře, je správa zdrojového kódu velmi propracovaná [48]. Samozřejmostí je hostování a správa neomezeného počtu privátních repozitářů se všemi základními funkcemi, které jsou dále rozšířeny zejména o pokročilou správu merge requestů, definici a vynucování politiky větví vč. zamykání jednotlivých souborů, podporu squash merge, fast-forward a rebase, propracované řešení konfliktů, podporu Git LFS a funkce pro code-review.

K dispozici jsou i pokročilé funkce pro propracovanou automatizaci na základě hloubkové analýzy kódu – např. detekcí klíčových slov TODO v kódu připravit tasky pro členy týmu, nebo automaticky provádět uzavírání relevantních tasků na základě přítomnosti klíčových slov v merge requestu. Mezi další zajímavé funkce patří propracovaná vizualizace grafu repozitáře a také unikátní WebIDE, které umožňuje provádět editaci kódu přímo z rozhraní SCM. Cloudová verze má limit na velikost jednoho repozitáře 10 GB.

4.3.3 Nástroje pro sestavování (build), CI / CD nástroje a automatické testování

Procesy CI/CD, včetně sestavování a testování, se v rámci GitLabu odehrávají odděleně od samotné instance GitLab serveru na tzv. runneru. Jako runner může sloužit jakýkoliv fyzický nebo virtuální stroj na kterém mohou buď běžet docker kontejnery, ve kterých se odehrává vlastní běh CI/CD úloh, nebo je běh realizován přímo s nástroji dostupnými v systému na kterém je runner instalován. Díky tomu je možné provádět sestavování, testování a jakékoliv další operace jakýmkoliv nástrojem, který je k dispozici formou docker image, přičemž to, jaký image se pro vykonání pipeline použije je definováno v konfiguraci pipeline.

Jediné zásadní omezení existuje pro non-Linux technologie v cloudové variantě GitLabu, kde jsou v rámci předplacených minut dostupné pouze Linux runnery (na spuštění Windows runnerů se v době psaní této práce pracuje). V případě využití proprietárních technologií jako je .Net Framework nebo Xcode je nutné instalovat runner na vlastní Windows, resp. Mac zařízení, což však díky oddělení GitLab serveru nebrání využívat je společně s cloudovou verzí GitLabu. Jednou ze zásadnějších slabin GitLabu je také absence širší podpory pro testovací frameworky. I když z principu fungování CI/CD neomezuje používání prakticky jakéhokoliv testovacího nástroje, dovede přehledně zpracovat výstupy pouze ve formátu JUnit XML, a to pouze základním způsobem. Pro podrobnější informace musí vývojář využít surového záznamu, zvláště při použití testovacích nástrojů, které daný uvedený výstupu generovat neumí.

GitLab též disponuje unikátní funkcí Auto DevOps, která má za cíl výrazně zjednodušit adopci CI/CD přístupů v jakémkoliv projektu [49]. Tento nástroj inteligentně zanalyzuje projekt a dle přítomných indicií v konfiguračních souborech se pokusí sám navrhnout celý CI/CD proces od sestavení, přes vytváření artefaktů a testování až po deployment. Tato funkce je v cloudové verzi zapnuta automaticky pro všechny projekty s tím, že se automaticky deaktivuje po prvním neúspěšném průchodu některé pipeline. I v takovém případě je však velmi užitečná, protože automaticky vytvoří základ všech procesů, který je pak možné pouze doladit pro správné fungování pro konkrétní potřeby projektu.

4.3.4 Podpora a zpětná vazba

GitLab obsahuje základní nástroj pro zajištění podpory koncových uživatelů nazývaný Service Desk. Principiálně pracuje tak, že každému projektu je vytvořena automaticky generovaná emailová adresa s tím, že každá nová příchozí zpráva založí příslušný issue v modulu řízení projektu, kde je s ním možné dále pracovat a to vč. automatického směrování další komunikace mezi zákazníkem a vývojáři přes e-mail. Tento přístup umožňuje jednoduše rozšířit správu projektu rozšířit o podporu pro koncové zákazníky, aniž by vývojáři museli opustit prostředí GitLabu. Tato funkce je však bohužel dostupná pouze ve vyšších variantách produktu (varianta Silver a vyšší u cloudové verze, resp. varianta Premium a vyšší u on-premise verze).

5. Výsledky a diskuse

5.1 Extrakce a normalizace vstupních dat

Na základě analýzy výše popsaných vlastností kandidátních řešení byly extrahovány konkrétní hodnoty jednotlivých kritérií.

U produktů jejichž cena je odstupňovaná v pásmech dle počtu uživatelů bylo nutné vzhledem k velkému počtu možných kombinací provést zobecnění, na základě kterého bylo vybráno několik reprezentativních variant pokrývající v obecné rovině možné potřeby cílových subjektů z kategorie malých podniků. Výběr variant byl proveden v souladu se zlomovými hodnotami pásem, které se navíc mezi jednotlivými produkty liší.

- Ekosystém Azure DevOps v cloudové verzi - 10, 20 a 50 uživatelů.
- Ekosystém Atlassian v cloudové verzi – 5 a 10 uživatelů
- Ekosystém Atlassian v on-premise verzi – 10, 25 a 50 uživatelů

Dále byla provedena normalizace dat, zejména s o hledem na skutečnost, že některá kritéria mohou nabývat neomezených hodnot. Tyto hodnoty jsou značeny kvalitativně matematickým symbolem ∞ a nabývají absolutní přednosti před ostatními variantami. Pro porovnatelnost byla cena za pravidelné platby přepočítána na jeden měsíc a jednoho uživatele. Všechny uvedené ceny jsou přepočítány na měnu CZK a to podle platného kurzu České národní banky. Výslednou sadu hodnot popisuje Tabulka 1 na následující straně.

Z této tabulky lze již nyní vyvodit určité dílčí závěry, díky kterým je možné optimalizovat další postup:

1. Je patrné, že některá kritéria splňují všechny kandidátní varianty. Jedná se o:
    - všechna kritéria skupiny Nástroje pro plánování a spolupráci,
    - všechna kritéria skupiny Automatické testování,
    - všechna kritéria skupiny Správa zdrojového kódu, s výjimkou kritéria Velikost úložiště.

    Ukázalo se, že podpora souvisejících DevOps procesů je všeobecně natolik dobrá, že přítomnost předmětných funkcí v této kategorii produktů je možné považovat za samozřejmost. Tato kritéria je tudíž možné zcela zanedbat, neboť nebudou mít žádný vliv na výsledky dalších postupů.

Výchozí, surová data získaná z ceníků jednotlivých ekosystémů ke dni 2.3.2019

Varianta	Cena (měsíc)/uživatel	Jednorázová platba	Max. uživatelů	Vlastní hosting	SCRUM	Kanban	Agile reporting	Git repozitář	E-mailové notifikace	Velikost úložiště (GB)	Revize zdrojového kódu	Vkládání komentářů	Akceptace merge požadavků	Velikost úložiště artefaktů v GB	Release management	CI/CD pipeline (min/měsíc)	Jednotkové testy	Testy komponent	Integrační testy	Systémové testy	Řetězové testy	Tiketovací systém	Reporty	E-mailové notifikace
	Obecná kriteria				Nástroje pro plánování a spolupráci			Správa zdrojového kódu						Build		CI/CD nástroje	Automatické testování					Podpora a zpětná vazba		
Azure DevOps Free	0,00 Kč	0,00 Kč	5	NE	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	2GB	ANO	1 800	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Azure DevOps 10	64,80 Kč	0,00 Kč	10	NE	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	2GB	ANO	1 800	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Azure DevOps 20	118,75 Kč	0,00 Kč	20	NE	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	2GB	ANO	1 800	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Azure DevOps 50	151,14 Kč	0,00 Kč	50	NE	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	2GB	ANO	1 800	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Azure DevOps Server	1 100,00 Kč	0,00 Kč	∞	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	2GB	ANO	∞	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
GitLab Free	0,00 Kč	0,00 Kč	∞	NE	ANO	ANO	ANO	ANO	ANO	10	ANO	ANO	ANO	1GB	NE	2 000	ANO	ANO	ANO	ANO	ANO	NE	NE	NE
GitLab Bronze	90,00 Kč	0,00 Kč	∞	NE	ANO	ANO	ANO	ANO	ANO	10	ANO	ANO	ANO	1GB	NE	2 000	ANO	ANO	ANO	ANO	ANO	NE	NE	NE
GitLab Silver	428,00 Kč	0,00 Kč	∞	NE	ANO	ANO	ANO	ANO	ANO	10	ANO	ANO	ANO	1GB	ANO	10 000	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
GitLab Gold	2 232,00 Kč	0,00 Kč	∞	NE	ANO	ANO	ANO	ANO	ANO	10	ANO	ANO	ANO	1GB	ANO	50 000	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
GitLab Core	0,00 Kč	0,00 Kč	∞	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	NE	∞	ANO	ANO	ANO	ANO	ANO	NE	NE	NE
GitLab Starter	90,00 Kč	0,00 Kč	∞	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	NE	∞	ANO	ANO	ANO	ANO	ANO	NE	NE	NE
GitLab Premium	428,00 Kč	0,00 Kč	∞	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	ANO	∞	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
GitLab Ultimate	2 232,00 Kč	0,00 Kč	∞	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	ANO	∞	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Atlassian Stnadrad 10	90,00 Kč	0,00 Kč	10	NE	ANO	ANO	ANO	ANO	ANO	10	ANO	ANO	ANO	1GB	NE	500	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Atlassian Standard 5	45,00 Kč	0,00 Kč	5	NE	ANO	ANO	ANO	ANO	ANO	5	ANO	ANO	ANO	1GB	NE	50	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Atlassian Server 10	0,00 Kč	902,00 Kč	10	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	ANO	∞	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Atlassian Server 25	0,00 Kč	160 304,00 Kč	25	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	ANO	∞	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Atlassian Server 50	0,00 Kč	295 583,00 Kč	50	ANO	ANO	ANO	ANO	ANO	ANO	∞	ANO	ANO	ANO	1GB	ANO	∞	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO

* Ceny a jednotlivá kritéria jsou platná k 2.3.2019
* Ceny jsou převedené na CZK dle platného kurzu, stanoveného Českou národní bankou ke dni 2.3.2019
* Neomezené prostředky jsou v případě on-premise řešení limitované velikostí vlastního hardware

Tabulka 1: Zdrojová data, zdroj: vlastní zpracování

2. Vyskytl se problém, kdy byla až v této fázi zjištěna existence variant, které kvůli absenci funkcionality nemohou být považovány za kompletní řešení pro DevOps, protože neumožňují realizaci určité části životního cyklu. Problém se týká ekosystému GitLab u variant Free a Bronze, resp. Core a Starter, kterým zcela schází funkcionalita pro zajištění podpory a zpětné vazby.

   I když jde jistě o použitelné a cenově zajímavé řešení, je absence podpory pro celou jednu část životního cyklu DevObs zásadní komplikací, protože je v důsledku nelze považovat za komplexní řešení, která tato práce zkoumá. Pokud by byly tyto varianty připuštěny v dalším postupu, zcela by se opustil kontext, v rámci kterého je zpracováváno porovnání, následkem čehož by toto ztratilo objektivní vypovídací hodnotu. Z toho důvodu je nutné tyto varianty z dalšího postupu vyřadit.

   Aplikací korekcí na základě těchto závěrů je možné sadu hodnot zjednodušit před dalším postupem. Sadu hodnot po aplikaci korekcí zachycuje Tabulka 2.

Varianta	Obecná kriteria				Správa zdrojového kódu	Build	CI/CD nástroje
	Cena (měsíc)/uživatel	Jednorázová platba	Max. uživatelů	Vlastní hosting	Velikost úložiště (GB)	Release management	CI/CD pipeline (min)/měsíc
GitLab Silver	428,00 Kč	0,00 Kč	100	NE	10	ANO	10 000
GitLab Gold	2 232,00 Kč	0,00 Kč	100	NE	10	ANO	50 000
GitLab Premium	428,00 Kč	0,00 Kč	100	ANO	∞	ANO	∞
GitLab Ultimate	2 232,00 Kč	0,00 Kč	100	ANO	∞	ANO	∞
Atlassian Stnadrad 10	90,00 Kč	0,00 Kč	10	NE	10	NE	500
Atlassian Standard 5	45,00 Kč	0,00 Kč	5	NE	5	NE	50
Atlassian Server 10	0,00 Kč	902,00 Kč	10	ANO	∞	ANO	∞
Atlassian Server 25	0,00 Kč	160 304,00 Kč	25	ANO	∞	ANO	∞
Atlassian Server 50	0,00 Kč	295 583,00 Kč	50	ANO	∞	ANO	∞
Azure DevOps Free	0,00 Kč	0,00 Kč	5	NE	∞	ANO	1 800
Azure DevOps 10	64,80 Kč	0,00 Kč	10	NE	∞	ANO	1 800
Azure DevOps 20	118,75 Kč	0,00 Kč	20	NE	∞	ANO	1 800
Azure DevOps 50	151,14 Kč	0,00 Kč	50	NE	∞	ANO	1 800
Azure DevOps Server	1 100,00 Kč	0,00 Kč	100	ANO	∞	ANO	∞

Tabulka 2: Očištěná vstupní data – varianty jednotlivých ekosystémů vyhovující DevOps

5.2 Obecné závěry ve vztahu k DevOps

Na tomto místě je vhodné zmínit několik obecných poznatků, které z výše uvedených zjištění vyplývají. Současný stav komplexních řešení pro řízení softwarového vývoje má vůči přístupu DevOps ambivalentní vztah.

Pro ilustraci k následujícímu textu viz také Obrázek 1. Na jednu stranu je zřejmá vynikající podpora fází životního cyklu spadající do první poloviny pojmu „Dev“ (tedy *plan, code, build a test*), pro které jsou k dispozici vyzrálé a prověřené technologie. Na stranu druhou však není ustálena podpora pro fáze z druhé poloviny životního cyklu „Ops“ (tedy *release, deploy, operate a monitor*) - nástroje pro sestavování a nasazení se mnohdy neobejdou bez různých berliček, podpora pro Windows a Mac platformy není zdaleka samozřejmostí, podpora pro monitorování a zpětnou vazbu nezřídka chybí zcela.

Vysvětlením pro tento nepoměr může být jednak skutečnost, že systémový přístup k těmto procesům je stále relativní novinkou, jednak značná diverzita souvisejících technologií, které dosud nedospěli k implementaci jednotných standardů, jako je tomu např. u správy zdrojového kódu, kterému zcela dominuje nástroj Git. Diskutovat příčiny tohoto stavu a možná řešení je spíše námětem pro budoucí práci na toto téma, nicméně zejména při výběru nástroj pro řízení vývoje je nutné mít tuto skutečnost na paměti již nyní vzhledem k očekávanému vývoji a vybírat řešení s ohledem na flexibilitu a dopřednou kompatibilitu.

5.3 Praktická ukázka aplikace modelu na konkrétní subjekt

Shromážděné informace jsou nyní již dostatečné k provedení praktické ukázky aplikací modelu na potřeby konkrétního subjektu, který je popsán v této sekci.

5.3.1 Charakteristika subjektu a jeho preference

Pro účely demonstrace zvolené metodiky byl na základě individuální dohody vybrán cílový subjekt. Jako součást podmínek spolupráce s autorem si subjekt vyhradil neuvést své jméno a svolil ke zveřejnění údajů pouze v obecné podobě. Získávání údajů od subjektu probíhalo formou nestrukturovaných rozhovorů s autorem práce.

Cílový subjekt je obchodní společnost působící ve Spolkové republice Německo. Podnikaná v oblasti poskytování software pro ubytovací zařízení, turistická informační centra a municipality. Z pohledu tuzemské kategorizace účetních jednotek by spadal do

kategorie mikro účetních jednotek, tj. do deseti zaměstnanců a čistého obratu do 18 mil. Kč.

Subjekt působí na trhu již přes deset let a v současné době spravuje portfolio vlastního software, který byl postupně během této doby vytvořen. Subjekt má stabilní klientelu nicméně hodlá rozvíjet svoje portfolio dále, v čemž mu nyní brání zejména zátěž legacy kódu. Motivací subjektu k hledání komplexního řešení DevOps je tedy především současný interní technický dluh, který pro něj nyní představuje zásadní bariéru v dalším rozvoji. Z toho důvodu velmi silně preferuje kritéria Release management a CI/CD pipeline, jelikož si uvědomuje nutnost jejich zavedení před další expanzí. Vzhledem k tomu, že subjekt již v současné době obhospodařuje jisté portfolio zdrojů (zdrojové kódy, artefakty a statický obrazový obsah) vnímá jako silně preferované kritérium na velikost úložiště, protože v případě nasazení jakéhokoliv řešení do něj hodlá migrovat všechna svá současná data navíc s perspektivou dalšího růstu na potřebu úložného prostoru.

Subjekt preferuje způsob nasazení on-site zejména z důvodu starosti jednak o své know-how, jednak o data jeho zákazníků s ohledem na místní legislativu a GDPR. I přesto by však pro něj cloudová varianta nepředstavovala problém, protože všichni poskytovatelé porovnávaných řešení mají implementováno compliance v dostatečné míře. Subjekt si je rovněž vědom skutečnost, že on-site nasazení by pro něj představoval dodatečné náklady. Z těchto důvodu je kritérium Vlastní hosting preferováno pouze slabě.

Subjekt rovněž nahlíží na kritéria nízké ceny jako na slabě preferované, zejména z toho důvodu, že si je vědom nutnosti změn svých interních vývojářských postupů a je ochoten do nich investovat přiměřenou část svého zisku.

Z technologického hlediska je pro subjekt zásadní podpora platformy .NET, na které je v současnosti založena většina jeho portfolia.

5.3.2 Výpočet a výsledek

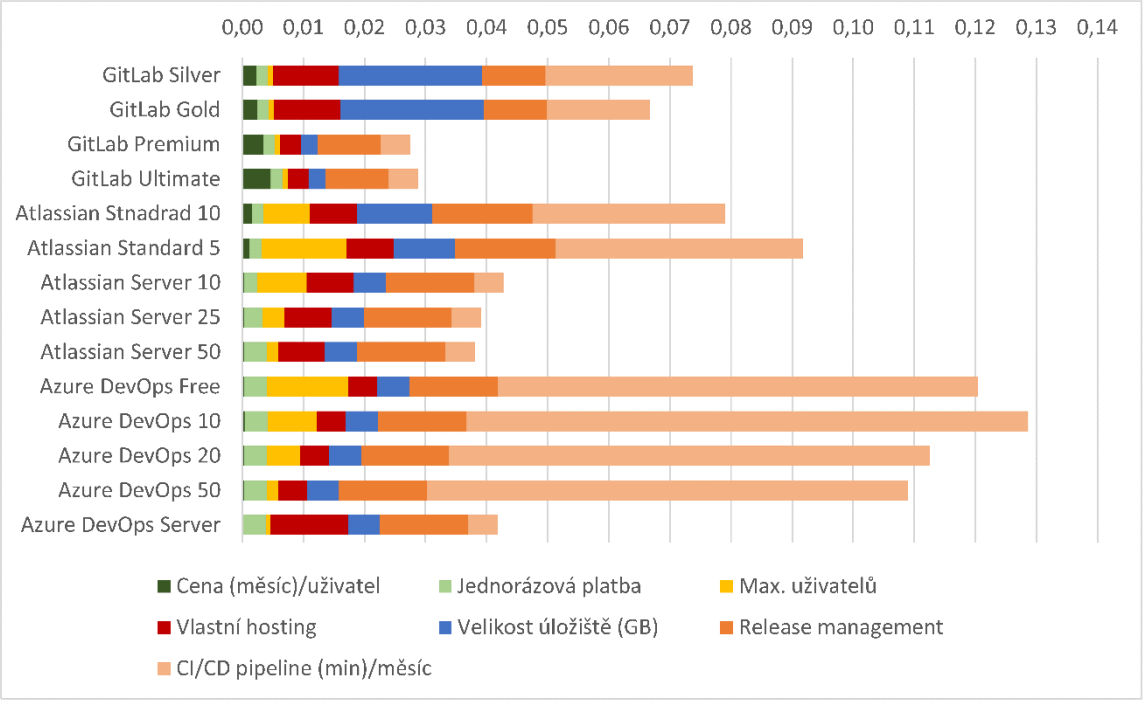
Na základě nestrukturovaných rozhovorů se subjektem, z nichž vyplynuly výše pospané preference, byla sestavena tabulka vah kritérií. Pro její sestavení bylo použito Saatyho metody, která je určena pro sestavování jedním subjektivně hodnotícím expertem. Výsledná matice vah je zachycena v Tabulka 3.

	Cena (měsíc)/uživatel	Jednorázová platba	Max. uživatelů	Vlastní hosting	Velikost úložiště (GB)	Release management	CI/CD pipeline (min)/měsíc
Cena (měsíc)/uživatel	1	0,11	0,20	0,11	0,30	0,11	0,14
Jednorázová platba	9	1	0,20	0,20	0,20	0,20	0,20
Max. uživatelů	5	5	1	0,30	0,30	0,30	0,20
Vlastní hosting	9	5	3	1	0,30	0,30	0,14
Velikost úložiště (GB)	3	5	3	3	1	0,30	0,11
Release management	9	5	3	3	3	1	0,11
CI/CD pipeline (min)/měsíc	7	5	5	7	9	9	1

Tabulka 3: Váhy kritérií sestavené na základě priorit konkrétního subjektu

Na základě této matice vah kritérií a očištěných vstupních dat (viz Tabulka 2) byly sestaveny preferenční tabulky porovnávaných variant pro každé kritérium. Jejich syntézou vznikla tabulka s výsledným hodnocením. Kompletní podrobnosti výpočtu vč. mezikroků jsou vzhledem k velkému rozsahu připojeny jako Příloha A této práce.

Pro lepší názornost byly výsledné preferenční hodnoty transformovány do podoby skládaného pruhového grafu, viz Obrázek 11.



Obrázek 11: Výsledek pro konkrétní subjekt ve formě skládaného pruhového grafu

Na základě těchto údajů vychází pro potřeby daného subjektu jako velmi vhodné všechny varianty Azure DevOps s tím, že jako optimální vyšla varianta Azure DevOps 20. Zásadní vliv kritéria CI/CD pipeline je očekávatelný vzhledem k udaným preferencím subjektu.

Pozoruhodné však je, proč byly silně penalizovány všechny on-premise varianty které z podstaty mají čas běhu CI/CD pipeline neomezený a bylo by možné očekávat, že právě ony budou nejvhodnějšími kandidáty?

Důvodem pro tento jev je zřejmě skutečnost, že všechny on-premise varianty jsou v porovnání s cloudovými řádově dražší a tudíž nevýhodné, a to i přesto, že subjekt cenové hledisko označil za slabě preferované. Tuto hypotézu však nepodporuje ekosystém GitLab, u nějž jsou ceny cloudové a on premise varianty zcela shodné. V tomto případě však sehrála roli skutečnost, že ekosystém Azure DevOps dává k dispozici neomezený úložný prostor, což je kritérium, které naopak subjekt označil za silně preferované.

Výše uvedený rozbor tedy potvrzuje, že výpočet zvolenou metodou vybral variantu, které je v souladu s požadavky subjektu.

6. Závěr

Hlavním cílem této práce bylo pomocí vícekriteriálních metod porovnat v současnosti dostupná řešení pro řízení vývoje softwarových projektů a navrhnout konkrétní řešení pro zvolený podnikatelský subjekt.

Nejprve byl v kapitole 3 na základě analýzy primárních a sekundárních zdrojů sestaven teoretický rámec, a to ve dvou rovinách – pro poznání potřeb vývojářů a hledání vhodných nástrojů v rovině jedné, a pro volbu vhodné metody porovnání v rovině druhé.

V první rovině byly analyzovány nejčastěji používané metodiky při vývoji software a diskutována zastřešující metodologie DevOps. Pro účely volby kandidátních řešení pro následné porovnání byly nejdříve identifikovány kategorie nástrojů používaných pro vývoj, jejich obecné charakteristiky a role ve workflow vývojářských týmů. Ve druhé rovině byly v sekci 3.4 diskutovány vícekriteriální metody porovnání, přičemž se jako vhodná ukázala Saatyho metoda. Rovněž zde byla definována konkrétní kritéria.

Se znalostí potřebného teoretického aparátu byla dále na základě předchozích zjištění a odkazované statistiky tržního podílu vybrána kandidátní řešení – ekosystémy Azure DevOps, Atlassian a GitLab. Jejich popis, zejména s ohledem na vlastnosti sledované kritérii zvolené porovnávací metody, je podrobně rozebrán v kapitole 4. Následně byla provedena v sekci 5.1 extrakce a normalizace hodnot kritérií z vlastností kandidátních řešení. Tím byl splněn druhý dílčí cíl. Nadto byla provedena korekce hodnot na základě obecných zjištění, která vyplynula při zpracování a která umožňují proces porovnání zjednodušit.

V tomto momentě již byl k dispozici dostatek dat pro učinění obecných závěrů v sekci 5.2. Následně bylo přistoupeno k jejich aplikaci na konkrétní subjekt, která je popsána v sekci 5.3. Na základě nestrukturovaných rozhovorů byly formalizovány preference subjektu a byla sestavena matice vah kritérií. Kombinací této matice a očištěných vstupních dat byly sestaveny preferenční tabulky pro každé kritérium a jejich syntézou vznikla výsledná tabulka a graf s hodnocením, přičemž jako optimální byla shledána varianta Azure DevOps 20. Výsledek porovnání byl následně podroben rozboru a byl konfrontován s preferencemi subjektu, přičemž byla konstatována shoda, a tudíž i splnění třetího dílčího cíle.

Na základě výše uvedeného je možné závěrem konstatovat že cíl práce byl splněn jak v části hlavní, tak v částech dílčích.

Seznam použitých zdrojů

1. WEBBER-CROSS, Geoff. *Learning Microsoft Azure*. Birmingham - Mumbai: Packt Publishing, 2014. ISBN 9781782173373.

2. JAMSA, Kris. Cloud computing: SaaS, PaaS, IaaS, virtualization, business models, mobile, security and more. Burlington, MA: Jones & Bartlett Learning, c2013. ISBN 9781449647391.

3. Ochrana osobních údajů: zákon o ochraně osobních údajů a další právní předpisy. GDPR - obecné nařízení Evropského parlamentu a rady (EU) 2016/679, o ochraně osobních údajů: redakční uzávěrka 28.8.2017. Ostrava: Sagit, [2017]. ÚZ. ISBN 978-80-7488-241-8.

4. VELTE, Anthony T, Toby J VELTE a Robert C ELSENPETER. Cloud Computing: praktický průvodce. Brno: Computer Press, 2011. ISBN 9788025133330.

5. WEBBER-CROSS, Geoff. Learning Microsoft Azure. Birmingham - Mumbai: Packt Publishing, 2014. ISBN 9781782173373.

6. ŠUBRT, Tomáš. Ekonomicko-matematické metody. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, 2011. ISBN 978-80-7380-345-2.

7. Co znamená DevOps? | atlassian. Atlassian | Software Development and Collaboration Tools [online]. [cit. 2018-04-05]. Dostupné z: https://cs.atlassian.com/devops

8. *Manifesto for Agile Software Development* [online]. 2010 [cit. 201-04-05]. Dostupné z: http://agilemanifesto.org/

9. VIJAYASARATHY, LEO R. a DAN TURK. AGILE SOFTWARE DEVELOPMENT: A SURVEY OF EARLY ADOPTERS. *Semantic Scholar - An academic search engine for scientific articles* [online]. COLORADO: COLORADO STATE UNIVERSITY, 2008, 2008 [cit. 2018-06-28]. Dostupné z: https://pdfs.semanticscholar.org/1d34/fadfeee3229a0e980a0030a67880c37c9450.pdf

10. HARDING, Alexander a Janet C. READ. A Study into the Adoption of, and Enthusiasm for Agile Development Methodologies Within Further Education. *AIS Electronic Library (AISeL) | Association for Information Systems Research* [online]. Cyprus: ISD2017 CYPRUS, 2017, 2017 [cit. 2018-06-28]. Dostupné z: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1148&context=isd2014

11. BEGEL, Andrew a Nachiappan NAGAPPAN. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. *ResearchGate: Share and discover research* [online]. 2007, 2007 [cit. 2018-06-28]. Dostupné z: https://www.researchgate.net/publication/4279044_Usage_and_Perceptions_of_Agile_Software_Development_in_an_Industrial_Context_An_Exploratory_Study

12. A 2017 Analysis of Agile Adoption Trends. *CodeBeamer Application Lifecycle Management: Intland Software* [online]. 15.09.2017 [cit. 2018-06-30]. Dostupné z: https://content.intland.com/blog/agile/a-2017-analysis-of-agile-adoption-trends

13. Agile 2008 Toronto: Agile Infrastructure and Operations Presentation. *Who or what is Jedi BVBA?* [online]. 2017 [cit. 2018-06-30]. Dostupné z: http://www.jedi.be/blog/2008/10/09/agile-2008-toronto-agile-infrastructure-and-operations-presentation/

14. BASS, Len, Ingo M. WEBER a Liming ZHU. *DevOps: a software architect's perspective*. New York: Addison-Wesley Professional, [2015]. ISBN 978-0134049847.

15. Has DevOps changed the role of a tester?. *Hacker noon* [online]. [cit. 2018-10-16]. Dostupné z: https://hackernoon.com/has-devops-changed-the-role-of-a-tester-b140dddc7824

16. Periodic Table of DevOps Tools Version 3 – Coming Soon. *XebiaLabs: Blog - Continuous Delivery & DevOps Tools for the Enterprise* [online]. 2018 [cit. 2018-12-23]. Dostupné z: https://blog.xebialabs.com/2018/04/12/periodic-table-of-devops-tools-version-3-coming-summer-2018

17. DevOps: Continuous Delivery, Integration, and Deployment with DevOps. Birmingham: Packt Publishing, 2018. ISBN 978-1-78913-299-1.

18. BIBIK, Ilya. How to kill the Scrum monster: quick start to agile Scrum methodology and the scrum master role. Berkeley, California?: Apress, [2018]. ISBN 9781484236901.

19. Easy Agile Scrum Workflow for Jira. *Atlassian* [online]. [cit. 2019-09-02]. Dostupné z: https://marketplace.atlassian.com/apps/1213278/easy-agile-scrum-workflow-for-jira?hosting=server&tab=overview

20. 4 ways to use GitLab Issue Boards | GitLab. *GitLab* [online]. 2018 [cit. 2019-09-02]. Dostupné z: https://about.gitlab.com/2018/08/02/4-ways-to-use-gitlab-issue-boards/

21. *Bitbucket* | *The Git solution for professional teams* [online]. Bitbucket, 2018 [cit. 2019-09-02]. Dostupné z: https://bitbucket.org/

22. DUVALL, Paul M., Steve MATYAS a Andrew GLOVER. *Continuous integration: improving software quality and reducing risk*. Upper Saddle River, NJ: Addison-Wesley, c2007. ISBN 978-0-321-33638-5.

23. AUGSTEN, Stephan. Was ist Continuous Integration?. *Dev-Insider - das Fachportal zu Software-Entwicklung und -Betrieb* [online]. 16.03.2018 [cit. 2019-10-07]. Dostupné z: https://www.dev-insider.de/was-ist-continuous-integration-a-690914/

24. AUGSTEN, Stephan. Was ist Continuous Delivery?. *Dev-Insider - das Fachportal zu Software-Entwicklung und -Betrieb* [online]. 12.08.2017 [cit. 2019-10-07]. Dostupné z: https://www.dev-insider.de/was-ist-continuous-delivery-a-664429/

25. GHAHRAI, Amir. Testing In DevOps - Strategies, Tools and Practices. *Testing Excellence — Software Testing for Beginners and Professionals* [online]. 2018, 02.12.2018 [cit. 2019-10-09]. Dostupné z: https://www.testingexcellence.com/testing-in-devops/

26. ARNAUT, Wagner. Continuous feedback in devops improves application deployment. *IBM* [online]. 13.06.2016 [cit. 2019-10-09]. Dostupné z: https://www.ibm.com/blogs/cloud-computing/2016/06/13/continuous-feedback-devops-application-deployment/

27. Co je Azure - Microsoft cloud services. *Microsoft Azure* [online]. 2019 [cit. 2019-10-10]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-azure/

28. Jira | Software pro sledování požadavků a projektů. *Atlassian | Software Development and Collaboration Tools* [online]. [cit. 2019-10-10]. Dostupné z: https://www.atlassian.com/cs/software/jira

29. *The first single application for the entire DevOps lifecycle - GitLab |* *GitLab* [online]. 2019 [cit. 2019-10-10]. Dostupné z: https://about.gitlab.com/

30. DevOps Artifacts - Artifactory, Sonatype Nexus, Maven Artifact Repository, and Apache Archiva - 2018. *Open Source: 2018 - Java/C++/Python/BigData/Android/* [online]. [cit. 2019-10-10]. Dostupné z: https://www.bogotobogo.com/DevOps/DevOps_Artifacts_Artifactory_Sonatype_Nexus_Maven_Artifact_Repository_Apache_Archiva.php

31. DevOps Test - Test types in the deployment pipeline. *Best Read on ITpedia* [online]. Holandsko, 2017-07-27 [cit. 2019-10-15]. Dostupné z: https://www.itpedia.nl/2017/07/27/test-types-in-the-deployment-pipeline/

32. Project Management Market Share Report | Competitor Analysis | Jira, Microsoft Project, Smartsheet. *The Leader in Technographics | Datanyze* [online]. San Mateo [cit. 2019-10-15]. Dostupné z: https://www.datanyze.com/market-share/project-management

33. Source Code Management Market Share Report | Competitor Analysis | Git, Github, Bitbucket. *The Leader in Technographics | Datanyze* [online]. San Mateo [cit. 2019-10-15]. Dostupné z: https://www.datanyze.com/market-share/source-code-management

34. Continuous Integration Market Share Report | Competitor Analysis | Jenkins, JetBrains TeamCity, CircleCI. *The Leader in Technographics | Datanyze* [online]. San Mateo [cit. 2019-10-15]. Dostupné z: https://www.datanyze.com/market-share/ci

35. Why Microsoft loves Linux | ZDNet. *Technology News, Analysis, Comments and Product Reviews for IT Professionals | ZDNet* [online]. [cit. 2019-10-15]. Dostupné z: https://www.zdnet.com/article/why-microsoft-loves-linux/

36. CONSTANTIN, Lucian. Attackers Up Their Game with Latest NPM Package Compromise - The New Stack. *The New Stack* [online]. 2015, 2018-11-30 [cit. 2019-10-15]. Dostupné z: https://thenewstack.io/attackers-up-their-game-with-latest-npm-package-compromise/

37. Azure DevOps Services | Microsoft Azure. *Cloudová výpočetní platforma a služby Microsoft Azure* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/

38. Review test results - Azure Pipelines | Microsoft Docs. *Technická dokumentace, rozhraní API a příklady kódování | Microsoft Docs* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://docs.microsoft.com/en-us/azure/devops/pipelines/test/review-continuous-test-results-after-build?view=azure-devops

39. Jira Pricing - Monthly and Annual Subscription Cost per User. *Atlassian | Software Development and Collaboration Tools* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://www.atlassian.com/software/jira/pricing?tab=cloud

40. Bitbucket - Features | Atlassian. *Atlassian | Software Development and Collaboration Tools* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://www.atlassian.com/software/bitbucket/features

41. Pricing | Bitbucket. *Atlassian | Software Development and Collaboration Tools* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://bitbucket.org/product/pricing?tab=cloud-tab

42. Bamboo – ceny | Atlassian. *Atlassian | Software Development and Collaboration Tools* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://www.atlassian.com/cs/software/bamboo/pricing

43. Jira Service Desk | IT podpora a vytváření ticketů. *Atlassian | Software Development and Collaboration Tools* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://www.atlassian.com/cs/software/jira/service-desk

44. Ceny – Jira Service Desk | Atlassian. *Atlassian | Software Development and Collaboration Tools* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://www.atlassian.com/cs/software/jira/service-desk/pricing?tab=self-managed

45. BIRMACHER, Barnabas. State of App Development in 2016 | Bitrise Blog. *Bitrise Blog* [online]. 2014, 2017-01-27 [cit. 2019-10-16]. Dostupné z: https://blog.bitrise.io/state-of-app-development-2016#self-hosted

46. GitLab.com Feature Comparison | GitLab. *The first single application for the entire DevOps lifecycle - GitLab | GitLab* [online]. [cit. 2019-10-16]. Dostupné z: https://about.gitlab.com/pricing/gitlab-com/feature-comparison/

47. GitLab Agile Delivery | GitLab. *The first single application for the entire DevOps lifecycle - GitLab | GitLab* [online]. [cit. 2019-10-16]. Dostupné z: https://about.gitlab.com/solutions/agile-delivery/

48. Product | GitLab. *The first single application for the entire DevOps lifecycle - GitLab | GitLab* [online]. [cit. 2019-10-16]. Dostupné z: https://about.gitlab.com/product/source-code-management/

49. Auto DevOps | GitLab. *GitLab Documentation* [online]. [cit. 2019-10-17]. Dostupné z: https://docs.gitlab.com/ee/topics/autodevops/

50. DRIESSEN, Vincent. Gitflow Workflow | Atlassian Git Tutorial. *Atlassian | Software Development and Collaboration Tools* [online]. [cit. 2019-10-17]. Dostupné z: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Přílohy

Příloha A CD s přiloženými soubory výpočtů v aplikaci Microsoft Excel