Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	LS 2018/2019
Autoři – jméno, příjmení, xname	Aneta Hubálková, xtura08 Daniel Tyrpekl, tyrd00 Michal Višňovský, vism00
Téma	Continuous Delivery
Datum odevzdání	19. května 2019

Abstrakt

Semestrální práce pojednává o přechodu z agilního vývoje a Scrumu na Continuous Delivery a Kanban ve firmě Rally Software. V práci je představena společnost Rally Software a následně vysvětleno, co Continuous Delivery je. Následuje popis praktik, které byly v Rally Software využity a v závěru práce shrnutí, která doporučení a praktiky se při přechodu na Continuous Delivery osvědčily, či naopak.

Klíčová slova

Continuous Delivery, Rally Software, Kanban

Obsah

Úvod

Při vývoji softwaru je možné se setkat s různými přístupy a metodikami vývoje – od tradičních rigorózních přes Scrum a Extrémní programování až po Kanban. Právě poslednímu zmíněnému, tedy Kanbanu, a přístupu zvanému Continuous Delivery, se věnoval článek, který je tématem této semestrální práce.

Cílem práce je kriticky zhodnotit přechod na Continuous Delivery ve firmě Rally Software. Téma této práce, Continuous Delivery, jsme si vybrali proto, že je stále vysoce aktuální – dle Amblera (“2013 IT Project Success Rates Survey Results,” n.d.) dosahovaly právě Lean metodiky, ke kterým se řadí i Kanban ve spojení s Continuous Delivery, nejlepších výsledků co se úspěšného dokončování projektů týče. Úspěšně dokončených produktů za pomoci Lean metodik dle tohoto průzkumu bylo přes 70 %, což je o necelých 10 procentních bodů více, než dosáhly další z metodik.

V semestrální práci bude nejdříve představena firma Rally Software, poté definováno Continuous Delivery, představeny praktiky použité firmou Rally při přechodu na CD, včetně například použité metodiky Kanban, a následně analyzován článek *Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy)* od Steeva Neelyho a Steeva Stolta z Agile Conference 2013. (Neely and Stolt, 2013)

1 Představení firmy Rally

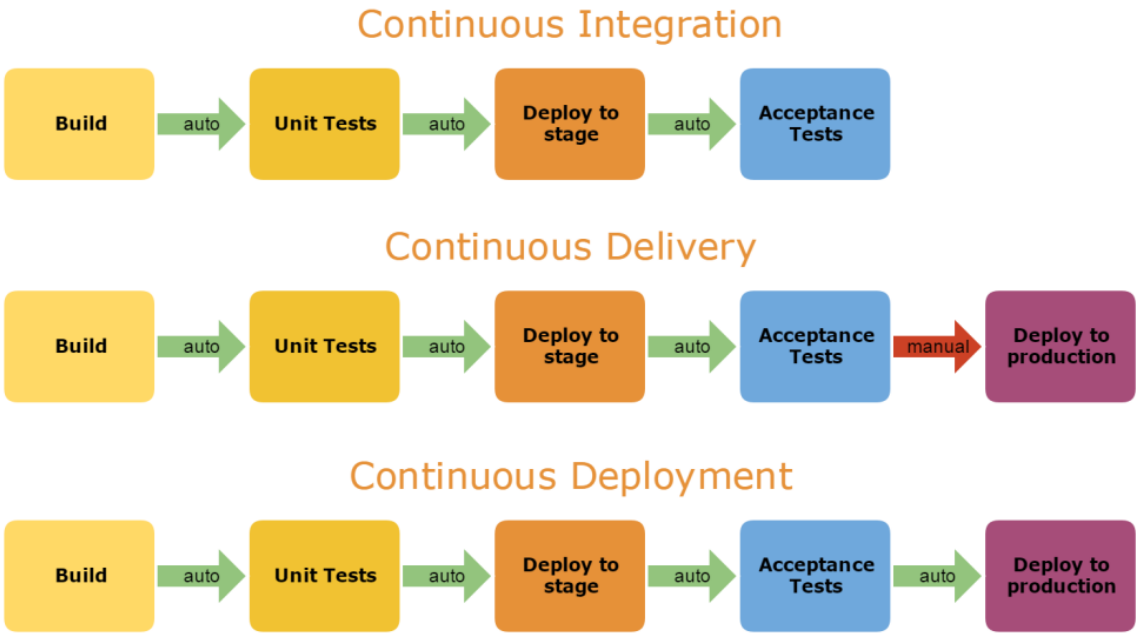
Článek pojednává o přechodu na Continuous Delivery ve firmě Rally Software. Rally Software Development je přední vývojářem software produktů pro agilní vývoj. Rally poskytuje také školící a konzultační služby ohledně Lean a Agile přístupů. (“Rally Software,” n.d.) V roce 2015 byly firma Rally koupena CA Technologies. (“CA Technologies acquires Rally Software - 2015-05-27,” n.d.)

Již při založení v roce 2002 firma praktikovala agile a scrum. Celých sedm let firma Rally praktikovala 8-týdenní Sprinty s výpadkovými sobotní releasy na konci Sprintu. V rámci neustálého zlepšování – continual improvement (neboli japonsky Kaizen) se firma rozhodla pro větší změnu, a to přechod ze Scrumu na Kanban a Continuous Delivery.

2 Continuous Delivery

Continuous Delivery patří do skupiny přístupu CI/CD, které jsou v poslední době zmiňovány v souvislosti s moderním vývojem čím dál častěji. Do rodiny CI/CD patří Continuous Integration, Continuous Delivery a Continuous Deployment.

Při Continuous Integration se často ukládané změny do repozitáře kontrolují pomocí automatických buildů a testů. Oproti tomu, při přístupu Continuous Delivery se takto vzniklý a otestovaný build rovnou připraví k nasazení na produkční prostředí, které však probíhá manuálně. Ještě o krok dále je přístup Continuous Deployment, při kterém probíhá release do produkce automaticky. Následující obrázek přístupy graficky porovnává. Stage je prostředí velmi podobné produkčnímu. (Source, 2017)



Obrázek I. – Continuous Integration versus Delivery versus Deployment (Source, 2017)

Všechny tyto přístupy používají CI/CD nástroje – nejčastěji Jenkins, nebo například GitLab (“20 Best Continuous Integration(CI) Tools in 2019,” n.d.), které jsou propojené s repozitáři (GitHub, Bitbucket), kde je uložen kód pomocí verzovacího systému (git, svn atd.). V CI/CD nástroji lidé z IT operations oddělení staví takzvané pipelines, kde jsou již předkonfigurované buildy, kterou jsou tím pádem zautomatizované a připravené na příští opakování. Programátor si tak usnadňuje provádění složitých a zdlouhavých kroků, jako je spouštění testů a zároveň může být například zaručeno, že sada testů proběhne, nebude ignorována a pokud testy neprojdou, nezdaří se ani build.

Aktuálně je místo přímo operations oddělení hojně využívaný DevOps přístup, který dnes přidává operační aspekty do každodenní práce developerů. Dalším pomyslným krokem za DevOps je ve firmách zakládání Site Reliability, takzvaných SRE Teamů, po vzoru Google, který s tímto konceptem přišel, a velmi se mu osvědčil. SRE teamy si stanovují takzvané error budgety, které se počítají dle sjednaných SLA – Service level agreements. (“Co to je SRE?,” n.d.)

2.1 Principy Continuous Delivery

Jelikož je velmi jednoduché ztratit se v realizaci Continuous Delivery, existují principy, které pomáhají začít se znovu soustředit na to, co je podstatné.

2.1.1 Vytváření kvality

Jednou z věcí, kde se dá ušetřit množství finančních prostředků je řešit problémy a závady, jakmile byly zjištěny. V ideálním případě jakmile byly zaznamenány a zjištěny po spuštění lokálních automatizovaných testech. Pokud se najde chyba po manuálním testu, je to časově mnohem náročnější a následně vyžaduje třídění defektu. Po určitém čase, kdy se chyba opravuje, je třeba si opět vzpomenout co bylo příčinou chyby. Někdy to může být řádově ve dnech nebo i týdnech. Pokud najdeme a následně hned opravujeme chybu, je třeba probrat, jak bychom to mohli zachytit automatizovanými akceptačními testy. Takto si umí organizace postupně budovat kvalitu svého výstupu.

2.1.2 Dělení celků na menší části

U tradičního postupu se zpětná vazba zjišťuje až po dlouhém čase, většinou až po testování. Při Continuous Delivery se postupuje opačně. Organizace se snaží dodat malou část co nejrychleji. Je to proto, aby získala zmíněnou zpětnou vazbu a v případě potřeby mohla reagovat na změny zadání. Rychlá zpětná vazba je jedna z mnoho výhod práce v malých dávkách. Další je i to, že usnadňuje třídění a opravu problémů, a v neposlední řadě zvyšuje efektivitu a motivaci. Jedním z důvodů, proč organizace pracují ve velkých dávkách je i určení fixní ceny za jeden celek. U Continuous Delivery je důležité změnit ekonomiku celého procesu dodání softwaru, například dělením na menší části.

2.1.3 Lidmi řešené problémy, počítači prováděné opakované úlohy

Aby se vzájemně doplňovaly počítače a lidé, je důležité správně rozdělit úkoly, které budou provádět. Určitě je lepší, když počítače budou pracovat na automaticky vykonávaných činnostech, které se opakují jako kdyby to dělali lidé. Může to být například u regresního testování. Počítačům nedělá problém vykonávat opakující činnost, ale řešit úkoly, které mají nepředvídatelný výstup, nemohou. Není cílem to, aby automatizace procesů vyřadila nebo nahradila lidi. Jde o zjednodušení v mnoha případech nudné opakující se práce. Proto jsou lidé zainteresováni do činností, které počítač nedokáže vyřešit sám. Dalším důvodů, proč počítače, a ne lidé, dělají opakovanou jednoduchou činnosti, je fakt, že lidé jsou právě při tomto druhu práce náchylní k chybám.

2.1.4 Neustálé zlepšování

Není dobré si myslet, že jsme se zlepšili na maximum a zastavit tento stav. Stále je co zlepšovat a být v něčem lepší ve srovnání s předchozím stavem. Pokud organizace něco zlepší, vždy bude existovat ještě alespoň jedna myšlenka, jak to zlepšit víc, hlouběji, lépe. Nejlepšími organizacemi jsou ty, které mají zaměstnance, kteří se chovají ke zlepšení práce jako k základní součásti své každodenní práce. Nemohou se spokojit s aktuálním statusem a přestat se zlepšovat, vyvíjet a vylepšovat.

2.1.5 Odpovědnost na každém z nás

Pokud chce být organizace výkonná, nesmí se problémy odkládat do neznáma nebo na někoho jiného. Například operační týmy odpovídají za pomoc vývojářům za budování kvality. Vývojáři odpovídají za stabilitu a kvalitu softwaru, na kterém pracují. Všichni zaměstnanci musí spolupracovat tak, aby spolu dosáhli cíle, který si na začátku stanovili. Na prvním místě je určitě dosažení cíle na organizační úrovni a až potom je optimalizace, která se týká pouze daného projektu nebo týmu. Stává se, že projekty snižují celkový výkon organizace tím, že mají špatné systémy na správu, špatné nastavené roční rozpočty, odměňování zaměstnanců atd. Může to být například odměňování testerů na základě počtu dokončených testů nebo nalezení chyb. U vývojáře to může být odměňování podle rychlosti psaní zdrojového kódu. Je velmi důležité pracovat na věcech, o kterých můžeme dostat rychlou zpětnou vazbu a zároveň má vysokou prioritu.

3 Důvody pro přechod na CD

Pro účely firmy Rally bylo Continuous Delivery zavedeno především pro možnost nasazování na produkci když je potřeba, s každým commitem. Nešlo ani tak o zvýšení frekvence releasů. Od začátku bylo jasné, že proces přechodu na CD bude velmi náročný po technické stránce. Všechny systémy musí být odlazeny, veškeré činnosti zautomatizovány, monitorovány a samozřejmě testovány.

Pro business znamenal původně nastavený development process s releasem jednou za osm týdnů dlouhá čekání na mnohdy velmi důležité funkce. Release byl pouze šestkrát do roku a byl plánován dlouho dopředu. S naplánovaným datem releasu však mohl marketing snáze odkomunikovat novou funkcionalitu všem stakeholders a především uživatelům, například pomocí načasování různých kampaní.

Důvody vývojářů jsou nesporné a je jich větší množství. V kódu, který se kupil celých osm týdnů, bylo nalezení případných problémů velmi složité a byly potřeba mnohdy rozsáhlé změny. Menší dávka kódu je tedy pro vývojáře mnohem výhodnější. Pro Continuous Delivery mluví také jednoduchý a rychlý rollback. Funkcionalita, která by se neosvědčila může být vývojářem jednoduše odstraněna. Vidět ihned výsledky své práce je důležité pro dobro celého týmu, proto je rychlejší feedback cycle (zpětná vazba) po krátkém Sprintu dalším krokem kupředu.

V neposlední řadě je důvodem pro CD také to, že vývojáři chtějí být prostě “cool”. Moderní úspěšná firma by měla ideálně zavádět nové technologie, frameworky a tím se pozvolna stát leaderem na trhu, díky iniciativě a přispění zapálených vývojářů. Ti pak o svých úspěších diskutují na blozích nebo například v podcastech.

4 Použité praktiky

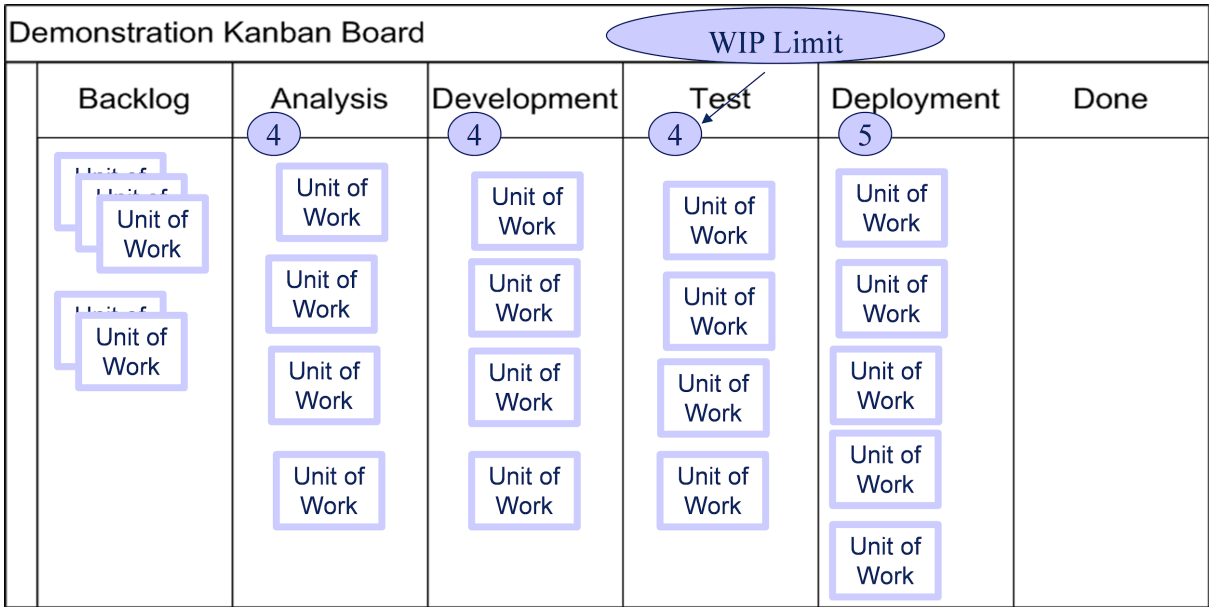
Přechod na Continuous Delivery vyžaduje na začátku mnoho úsilí. Je možné použít různé metodiky nebo provádět činnosti, které pomohou zmírnit dopady tohoto přechodu.

4.1 Kanban

Kanban je metodou pro snazší řízení zlepšování pracovních procesů v organizacích. Koncept byl vyvinut v Japonsku primárně pro automobilový průmysl, ale rozšířil se i do dalších odvětví. Vychází z principů štíhlé metody, tedy Lean Software Development.(“the definition of kanban,” n.d.)

Kanban je pojmenován po japonském výrazu “kamban”, který doslova znamená “lístek” nebo “karta”. Měl by zajišťovat časovou optimalizaci celého procesu od začátku až do konce a je k tomu využíváno častými vizualizacemi, typicky tzv. Kanban Boardem (zde je vidět, proč byl kanban pojmenován právě takto).

Od Scrumu se Kanban vyznačuje typicky tím, že pro jeho fungování nejsou potřeba žádné role a běží plynule bez jakýchkoli iterací.



Obrázek II. – Kanban Board (Hybridization et al., 2013)

4.2 Přechod z 8-týdenního nasazování na CD

Rychlý přechod přímo z 8-týdenních cyklů nasazování do strategie "push to production" v žádném případě není dobrý přístup. Společnost začala nejprve postupně zmenšovat jejich cykly nasazování na dvoutýdenní, týdenní, půl týdenní až po nasazování "at-will" (po vůli/kdykoliv). Uskutečnění těchto změn trvalo týdny, ba dokonce až měsíce. Za toto období se uskutečnilo mnoho přípravných prací na zefektivnění a zautomatizování procesu nasazování.

4.3 Dokumentování procesu

Společnost strávila hodně času tím, že zdokumentovala celý svůj proces. Tento proces mám na mysli od vytvoření userstorky až po nasazení na produkci.

4.4 Funkce přepínání kódu

Pro správu změn ve zdrojovém kódu je dobře používat GIT. Mnoho organizací si tento kvalitní nástroj chválí. Vývojáři pracují na vlastních věcech mimo hlavní větev s názvem "master", kde si mohou upravovat kód, aniž by ovlivnili práci jiným vývojářům. Problém nastává až tehdy, když chtějí své kódy sjednotit. Tam právě mohou vznikat "merge" konflikty, kde vyřešení může trvat v některých případech i několik hodin. Při nepozornosti může konflikt rozbít větev master,

což přináší další problémy. Ačkoliv GIT dokáže slučovat kód velmi dobře, i při malé chybě je pak velmi obtížné najít viníka.

Některé organizace proto začaly namísto větví využívat přepínače funkcí kódu. V podstatě jde o vytvoření podmíněné logiky, která skryje část změněného kódu, který ještě není připraven k nasazení. V administrativním rozhraní můžeme určit kde a kdo uvidí tuto funkci. Dokážeme monitorovat kód a zjistit vliv na systém. V případě chyby můžeme funkci vypnout a pokračujeme starou částí, která je nadále plně funkční. Jediné, na co je třeba dát při přepínání pozor je to, že si musíme být vědomi, co bude zapnuto současně na produkci.

4.5 Nasazení kanárků

Dark deployments je způsob, kdy se testují služby aplikace na reálném provozu. Snižuje riziko nenalezení bugů, které by testovací provoz nemusel objevit. Organizace vydá novou verzi pro malou část uživatelů a čeká na zpětnou vazbu. Pokud bude vše v pořádku, nasazuje se verze i pro další uživatele.

Canary deployments je tzv. nasazení na produkci a následné testování nového kódu, řekněme "testovacího" kódu, který můžeme rollbacknout, kdyby došlo k problémům. Metafora je převzatá z "the canary in a coal mine". Horníci používali kanárky na zkoumání dolů. Pokud se kanárek vrátil bylo bezpečné pokračovat v kopání, pokud se nevrátil znamenalo to, že nebude bezpečné pokračovat dál. Tento kód můžeme nazvat "kanárek". Pokud kanárek zemře, tak dáme tento nový kód ze systému pryč. Pokud nasadíme do systému kanárka, tak musíme zajistit, aby všechny modifikace, které provede na sdílených datech, byly zpětně kompatibilní se zbytkem systému.

4.6 Plánování testů

Pokud se organizace rozhodne nasazovat na produkci častěji, bude muset odstranit nebo značně omezit, manuální testování v krocích. Předtím se na manuální testování organizace spoléhali, a to hlavně z důvodu, že zkušení testeři dokážou najít místa, která jsou náchylnější na chyby. Toto právě zdržuje nasazování v kratších časových úsecích. Ani teď se nezapomíná na fázi vyčíslení odhadů testování. Zde spolupracují vývojáři a testeři, kteří dokumentují úplný seznam automatizovaných testů. Postupně organizace dokáže pokrýt programovací kód automatizovanými testy a vývojářům uzavřít mezery chybovosti. Tím pádem tým dostává jistotu, že daná část aplikace byla prověřena automatizovanými testy, a je v pořádku. Výsledkem plánování testů byl fakt, že již nebude nutné ruční testování většiny aplikace pro najití regresních chyb.

4.7 Představení nových funkcí

Když organizace nasazovala nové věci na produkci každých 8 týdnů, vždy předtím dávalo smysl nové funkce představit. Při postupném nasazení na produkci už nedávalo smysl vždy představovat na velkém meetingu malé nepatrné funkce. Tyto malé části dokonce nemusí být viditelné pro všechny uživatele. Řešení našli v tzv. "Přepínání funkcí". Je to možnost zapnout resp. vypnout na přání určitou část funkcí pro konkrétní uživatele. Mohou novou nepatrnou funkci spustit pro interní zákazníky, beta uživatelů nebo nastavit "rollback" pokud by došlo k nepředvídatelnému negativnímu vedlejšímu efektu. Po čase je možné více funkcí spustit už pro všechny uživatele. Jelikož se nevyplatilo vytvářet meetingy každý týden na představení nových funkcí, byl vymyšlen speciální přístup. Schůzky byly nahrazeny e-maily a informačními články na interních blozích.

4.8 Kde jsou moje data?

Jedna z chyb, kterou nám mohla odhalit hlubší analýza prováděna ve všech odděleních, byla nepřipravenost oddělení pro rychlé nasazení nových funkcí. Byl to zejména marketing a prodej. Obě oddělení spoléhala na 8-týdenní nasazení nových funkcí, protože jim to dávalo čas na přípravu. Nejhorší na tom byl člověk s rolí "product owner". Právě na toho přicházely otázky a žádosti, kde bylo požadováno více informací o čase a specifikaci nových funkcí. Product owner to přesně nevěděl a zároveň nechtěl šířit nepravdivé nebo neověřené informace. Nakonec se dozvěděli, že stačí zvýšit míru transparentnosti s prodejem a marketingem. Šlo o to, aby oddělení mělo alespoň představu o nových plánovaných funkcích.

5 Lessons Learned

Přechod ze Scrumu a 8 týdenních sprintů na Kanban a Continuous Delivery vyžadoval v Rally Software svůj čas, mnoho testování a samozřejmě ho vedle úspěchů a okamžitých zlepšení provázela také různá selhání, ze kterých si každý tým odnášel různé poznatky a poučení, co dělat, nebo naopak nedělat, jakým způsobem přistupovat k různým procesům a jak nakonec dosáhnout co největší efektivity veškerých aktivit s Continuous Delivery spojených.

5.1 Testování

První důležitý poznatek se týkal testování. Když se v Rally Software pokoušeli o deployment významné části kódu, neustále selhávaly testy a deployment nebyl úspěšný. Domnívali se, že

je to způsobeno chybnými daty, na kterých byl kód testován. Proto, aby se jim podařilo projít přes všechny testy, byla data odstraněna a kód byl spuštěn do produkce. Bohužel vše ale přestávalo fungovat. Posléze bylo zjištěno, že problém nebyl v testovacích datech samotných, ale v tom, že nový kód nebyl kompatibilní s jejich datovým formátem. Daný kód byl spuštěn do testovací produkce a naštěstí bylo ovlivněno jen několik beta zákazníků.

Díky této skutečnosti v Rally Software zjistili, že každý test má svůj důvod a nikdy by se neměly testy vynechávat či přeskakovat pouze na základě nepotvrzených domněnek. Pokud už by se s testy mělo nějakým způsobem manipulovat, vždy je důležité porozumět tomu, jaké z toho mohou vyvstat důsledky a i když je důležité vydávat nové verze softwaru poměrně rychle, nikdy by to nemělo být na úkor překakování testů a následnému potýkání se s tím, že celý software přestane fungovat.

5.2 Work-life balance

Testů se týká i druhé poučení. Díky velmi obsáhlé síti automatizovaných testů, ke kterým je přistupováno s určitou důležitostí a respektem, je release nových verzí softwaru mnohem rychlejší a bezpečnější, než tomu bývalo u 8-týdenních sprintů. Dříve měli v Rally Software tzv. Scary Saturdays – zakončení sprintu tím, že se v jednu sobotu museli všichni sejít a podílet se na releasu nové verze softwaru, kdy se vše mohlo ještě pokazit.

Při přechodu na Continuous Delivery se testovalo v průběhu celého vývoje a vzhledem ke skutečnosti, že přechod na novou verzi softwaru mohl proběhnout prakticky kdykoliv, a ne již v předem určenou sobotu, byly všechny týmy šťastné, že nemusí trávit víkendy v práci, ale na softwaru se opravdu pracuje pouze v pracovní době.

5.3 Monitoring

Další důležitou lekcí bylo to, že pro rychlé a časté deploymenty nových verzí softwaru je velmi důležitý monitoring veškerých procesů, které ve vývoji probíhají. Každý člen týmu by měl sledovat a monitorovat to, kdy je naplánovaný další release nové verze, zejména proto, aby věděli, jak zrovna jejich kód může novou verzi ovlivnit. Zároveň se snažili držet pravidla, že rollback bude až tím posledním využitým řešením, hlavně díky tomu, že za dobu mezi commitem a rollbackem už mohou být další verze commitu.

5.4 Onboarding - nábor nových zaměstnanců

V Rally Software také přišli na to, že díky již zmiňované obsáhlé síti automatizovaných testů je mnohem jednodušší pro nové členy týmu začít pracovat bez starosti a strachu na nových verzích softwaru. Testy zajišťovaly bezpečnost celého systému a vývojáři se tak nemuseli obávat toho, že právě jejich kód by mohl v softwaru napáchat nějaké škody.

5.5 Mindset

Každý člen týmu se po přechodu na Continuous Delivery musel také potýkat s tím, že měl mnohem větší zodpovědnost za to, aby své činnosti prováděl s mnohem větší opatrností – typicky vývojáři, kteří, jak už je zmiňováno výše, měli zodpovědnost za monitoring nejen svých činností, ale i činností dalších členů týmu, stejně tak museli být důslední ve sledování testů.

Některým týmům se mohla také pozměnit náplň práce – např. v článku zmiňovanému Quality Assurance týmu. V první fázi přechodu na Continuous Delivery se obávali toho, že nebudou mít čas na to, aby před deploymentu nové verze softwaru veškeré součásti otestovali. Obavy byly ale zbytečné. Museli se naučit důvěřovat celé síti automatizovaných testů, které práci prováděli za ně. Díky tomu se poté ale obávali toho, že z celého procesu vývoje budou „vyšachováni“. To samozřejmě také nebyla pravda. Jejich náplň práce nyní spočívala v tom, že museli testy důkladně plánovat, procházet backlogy a brainstormovat ohledně různých pohledů a přístupů k testování.

Podobně také tým Operations se musel potýkat se změnami. Doposud byli zvyklí na přesně naplánované datum releasu nové verze, kdy spouštěli startup skripty. To se díky Continuous Delivery změnilo a jejich hlavní náplní práce byla podpora procesu Continuous Delivery díky SSL tunelům a monitoringem různých řešení.

5.6 Paralelizace testů

Dalším problémem, který v rané fázi přechodu týmy zažívaly, byl příliš dlouhý chod testů, v tomto případě to bylo konkrétně 9 hodin. To bylo pro týmy ale příliš dlouho a problémem se stávalo časté ignorování testů, což je jeden z problémů zmíněných již výše.

Řešením se ukázala být paralelizace testů. Testy byly rozčleněny dle délky běhu, ty delší a náročné byly vyčleněny, upravovány a optimalizovány, a mezitím se spouštěly ty kratší. Bylo tím dosaženo mnohem kratší doby testování, tedy mnohem kratší doby potřebné k tomu, aby mohl proběhnout nový deployment.

5.7 Dokončování procesů

Zajímavým řešením proto, aby byla optimalizována doba releasu, které se ujalo konkrétně v oddělení Engineeringu, bylo tzv. „způsobování bolesti“. Konkrétně šlo o led pásky, které se po průběhu testů rozsvítily buď zeleně, nebo červeně. Zelená znamenala, že všechny testy proběhly v pořádku a že se může nová verze spustit do oběhu. Červená znamenala to, že minimálně jeden test selhal a je potřeba danou část kódu opravit.

Toto řešení nutilo vývojáře ke stále rychlejší optimalizaci kódu a celkových procesů spojených s releasem nových verzí. K ještě větší podpoře byla napsána malá aplikace, která zaznamenávala statistiky o tom, jaké jsou četnosti rozsvěcování červené či zelené barvy, jak dlouho trvá chyby opravit apod.

5.8 Business plány

Poslední v článku zmíněnou změnou, která díky přechodu na Kanban a Continuous Delivery proběhla, je to, jakým způsobem bylo přistupováno k business plánům.

Produktový tým má samozřejmě stále svůj plán na nadcházející období, ale momentálně to vypadá tak, že už není detailně rozpracovaný na celý rok, jako tomu bývalo, ale nyní je detailně rozpracován následující kvartál, na ten další jsou definovány změny, které by se měly udát, a na nadcházející půlrok jsou v plánu zmíněné možné změny, ze kterých by se mělo čerpat při detailní tvorbě nového plánu na další kvartály.

Zároveň je v odděleních časová osa, která všem zainteresovaným stranám říká, co se bude v následujících týdnech řešit a kdo z týmu na čem přesně pracuje. I přes rychlé a časté releasy je tímto tedy dosaženo skvělé úrovně transparentnosti. Ta je podpořena i na pravidelných měsíčních poradách, kde se průběh vývoje a další progress diskutuje.

Závěr

Článek, který je v semestrální práci rozebrán, se týkal firmy Rally Software a popisoval, jak společnost přešla ze Scrumu a 8-týdenních sprintů na Kanban a tzv. Continuous Delivery, neboli release nových verzí rychle, plynule a prakticky kdykoli.

CD staví na důvěře a transparentnosti mezi stakeholders. Pro všechny zainteresované strany je naprosto klíčové zcela porozumět procesu a poté začít s automatizací těch nejpomalejších manuálních činností nasazování. V průběhu přechodu na Continuous Delivery se v Rally Software nevyvarovali různých chyb a selhání, o to cennější ale poté bylo jejich poučení z nich. Potřeba investovat čas a peníze do automatizovaných testů je nevyhnutelné a je potřeba specializované Quality Assurance specialisty. Důležité je stavět na řádném test plánu, který sestavuje QA společně s developery ve dvojicích, případně trojicích. Vymýšlejí kde přidat automatizované testy a také hlídají test coverage, tedy procento pokrytí kódu unit testy.

Z měřených metrik společnosti Rally vyplynulo, že po přechodu na Continuous Delivery se zlepšila tzv. velocita, tedy story points se v průběhu sprintu zvýšily. Taktéž počet klienty reportovaných defektů na produkci se snížil. Celkový přínos pro společnost byl po přechodu rozhodně kladný, místo 8-týdenních sprintů mohou vydávat nové verze SW kdykoli chtějí, procesy pro tyto skutečnosti se jim podařilo zoptimalizovat a výkonnost celého systému se díky přechodu na Continuous Delivery a Kanban zvýšila.

Literatura

20 Best Continuous Integration(CI) Tools in 2019 [WWW Document], n.d. URL https://www.guru99.com/top-20-continuous-integration-tools.html (accessed 5.13.19).

2013 IT Project Success Rates Survey Results [WWW Document], n.d. URL http://www.ambysoft.com/surveys/success2013.html#Results (accessed 5.19.19).

CA Technologies acquires Rally Software - 2015-05-27 [WWW Document], n.d. . Crunchbase. URL https://www.crunchbase.com/acquisition/ca-acquires-rally-software--0b85c8fd (accessed 5.10.19).

Co to je SRE? [WWW Document], n.d. URL https://blog.prskavec.net/2016/03/10/co-to-je-sre/ (accessed 5.17.19).

Hybridization, S.P. of, Process, D.P.T.| S., Says, M., 2013. Kanban: Process Improvement and Bottlenecks. Software Process and Measurement. URL https://tcagley.wordpress.com/2013/09/12/kanban-process-improvement-and-bottlenecks/ (accessed 5.19.19).

Continuous Delivery. Principles - Continuous Delivery. URL https://tcagley.wordpress.com/2013/09/12/kanban-process-improvement-and-bottlenecks/ (accessed 5.18.19).

Neely, S., Stolt, S., 2013. Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy), in: 2013 Agile Conference. Presented at the 2013 Agile Conference (AGILE), IEEE, Nashville, TN, USA, pp. 121–128. https://doi.org/10.1109/AGILE.2013.17

Rally Software [WWW Document], n.d. . Crunchbase. URL https://www.crunchbase.com/organization/rally-software (accessed 5.10.19).

Source, E., 2017. Devops: Continuous Integration vs Continuous Delivery vs Continuous Deployment. Nastel Technologies, Inc. URL https://www.nastel.com/blog/devops-continuous-integration-vs-continuous-delivery-vs-continuous-deployment/ (accessed 5.17.19).

the definition of kanban [WWW Document], n.d. . www.dictionary.com. URL https://www.dictionary.com/browse/kanban (accessed 5.17.19).