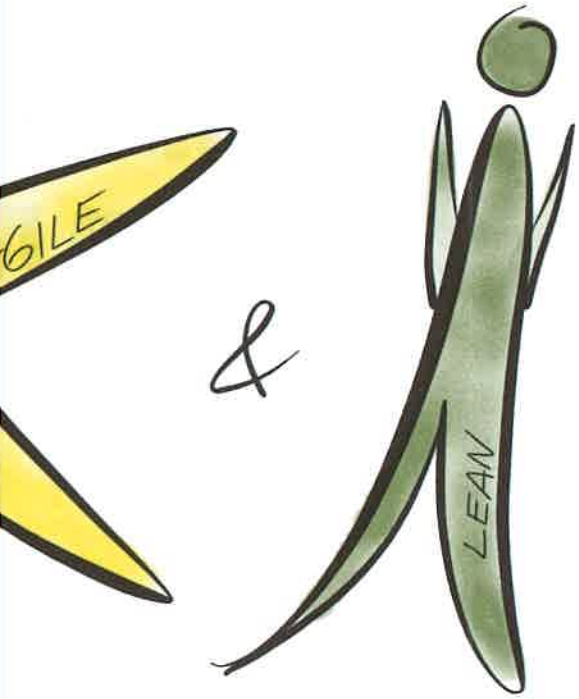


Agilní a lean metody

Co si představíte pod slovem „agilní"?

Agilní je dynamický, rychlý, interaktivní, přizpůsobivý, iterativní, zábavný, hravý, rychle reagující na změnu, … a jistě vymyslíte spoustu dalších synonym. Je to jiný způsob života, upřednostňující jiné hodnoty. Reálný výsledek před striktními procesy, změnu před předem naplánovaným. Být agilní znamená žít agilní filozofií, je to odlišná firemní kultura a nálada. Ale jestli čekáte kuchařku, jak se stát agilní, budete zklamaní. A jestli očekáváte nějaký model či framework s několika stupni agility, nebo dokonce checklist, tak vás bohužel zklameme. Nic takového nehledejte, protože to ani neexistuje, ač mnoho certifikovaných hlav tvrdí pravý opak. Žádný certifikát agility vám v pochopení nepomůže, maximálně může nastartovat proces přeměny. Ale agilní musíte být, agilně musíte myslet, agilně se chovat.

Ale nebojte se, není to zas tak neuchopitelné, jak se na první pohled zdá. Agile je o spolupráci a komunikaci a připravenosti na změnu. O tom, že zásadně děláme to, co má v danou chvíli smysl, a děláme to tak, jak nejlépe umíme. Agile sice není žádný striktní proces, ale není to ani žádný chaos. Má svá jasná pravidla. Dalo by se říct, že definuje hranice a vytyčuje menší hřiště, v jejichž rámci si týmy mohou stanovovat svá vlastní pravidla hry, tak aby se jim dobře pracovalo a byly co nejvíce produktivní, efektivní a dodali kvalitní produkt v co nejkratším čase. Takový tým je zaměřený na business value, tedy hodnotu pro zákazníka. Na to, jak optimalizovat funkcionalitu tak, aby zákazník byl maximálně spokojený a dostal za vynaložené prostředky to, co opravdu potřebuje a může používat.

Základním stavebním kamenem je – jak jinak – manifest; v tomto případě přímo agilní manifest, který shrnuje ve čtyřech bodech, co to vlastně znamená být agilní.

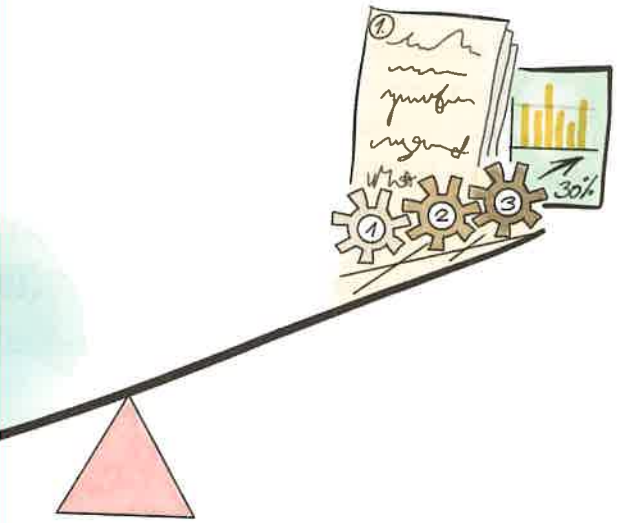
Manifest agilního vývoje softwaru

http://agilemanifesto.org

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

- ☐ Jednotlivci a interakce před procesy a nástroji
- ☐ Fungující software před vyčerpávající dokumentací
- ☐ Spolupráce se zákazníkem před vyjednáváním o smlouvě
- ☐ Reagování na změny před dodržováním plánu

ů nalevo si
nyslet nad

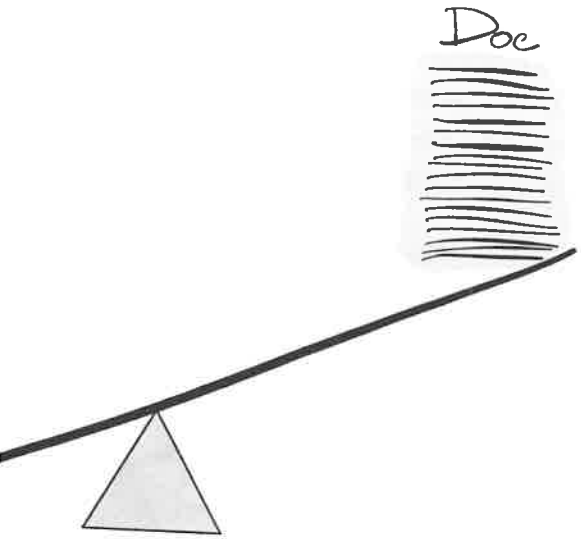


Jednotlivci a interakce před procesy a nástroji

Je známým faktem, že spolupracující týmy mají lepší výsledky než skupiny individuálně pracujících jednotlivců. Procesy a nástroje jim pomáhají dosáhnout výsledků, ale nejsou pro jejich úspěch nijak klíčové. Pojďme se podívat na dva odlišné týmy. V prvním máme 10 profesionálů, kteří si vytvořili vlastní nástroje a pracují v silně kolaborativním prostředí, ve druhém máme 10 vývojářů, kteří pracují v přesně definovaném procesním prostředí s nejlepšími nakoupenými nástroji, ale bez interakcí a spolupráce.

Co myslíte, který z těchto týmů vytvoří lepší software v kratším čase? Myslím, že se shodneme na tom, že první tým má větší šanci uspět, protože vzájemná spolupráce a komunikace mnohokrát převáží přesně definované procesy a nástroje. Ostatně podívejme se na úspěšné startupy, kde moc procesů nenajdeme, zato všichni překypují spoluprací a komunikací. No, a hlupák i s nejlepším nástrojem je zkrátka pořád hlupák.

Na druhou stranu, manifest neříká nic o tom, že procesy a dohody by neměly existovat ani že by týmy měly pracovat zcela bez nástrojů. Jen by měly mít možnost si nástroje vybrat a používat jen ty, které jim opravdu pomáhají v dosažení kvalitního výsledku.



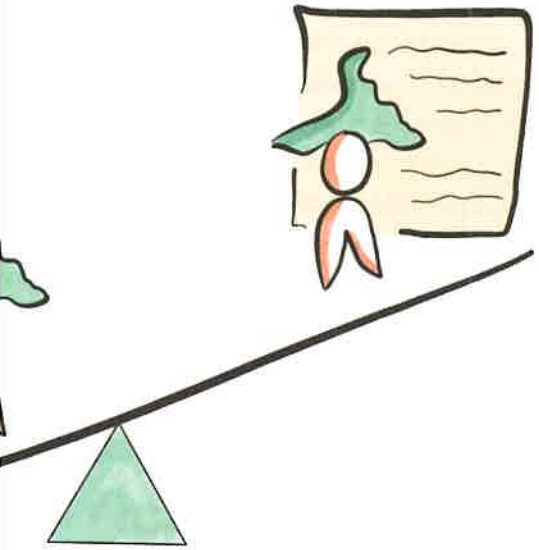
Fungující software před vyčerpávající dokumentací

Když se zeptáte zákazníků, zda by si koupili dokument popisující váš popis plánovaného produktu s architektonickou dokumentací, nebo ten samý produkt s minimální dokumentací, co si vyberou? Nebo praktičtější příklad: když si jdete koupit nový mobilní telefon, vyberete si pouze podle technické specifikace na papíře, nebo bude pro vás důležité si vyzkoušet i vlastní telefon před tím, než si ho koupíte? A když si koupíte nový televizor, přečtete si nejdříve manuál, nebo chcete vyzkoušet, jak funguje, a manuál čtete až na posledním místě? Lidé (ano, i zákazníci jsou lidé) zkrátka preferují praktické seznamování s produktem, předteoretickým.

Mnoho softwarových firem na toto zapomíná a chce ohromit zákazníka množstvím dokumentace, ale když dojde na vlastní demo, výsledek často dokumentaci odporuje a zákazník je z „funkčního" softwaru poněkud překvapený. Dokumentace **je** důležitá, ale neměla by převážit nad vlastním produktem; měla by primárně sloužit jako reference pro oblasti, které nejsou intuitivní a snadno pochopitelné. A takových oblastí by mělo být minimum.

Interní dokumentace se často píše pro budoucí týmy, aby se znalost architektury produktu neztratila. To samo o sobě není nijak špatný nápad, ale už jste někdy zkoušeli něco dohledat v deseti úrovních dokumentů, každý o 1000+ stranách? Vyčerpávající. A když už si dáte tu práci a prohledáte tu kupu slov, v devíti z deseti případů zjistíte, že daná věc v dokumentaci chybí či je v poslední verzi již zcela jinak. Interní dokumentace by měla být stručná a postihnout klíčové informace. Ostatní je obvykle efektivnější dokumentovat přímo v kódu.

Poslední případ je dokumentace funkcionalit v průběhu vývoje, kterou můžete v podstatě úplně odstranit a nahradit dobrou komunikací mezi analytiky, testery a vývojáři. Ti se pak sami domluví, co je třeba zdokumentovat pro ‚budoucí generace' vývojářů a testerů a v jaké podobě. Na závěr připomeneme, že nedoporučujeme přestat dokumentovat, ale jen dokumentaci omezit na minimum, aby poměr mezi námahou a časem, který investujete do psaní dokumentace, odpovídal hodnotě, kterou její zákazníci, tedy všichni, kteří ji kdy čtou, z takovéto dokumentace načerpají.

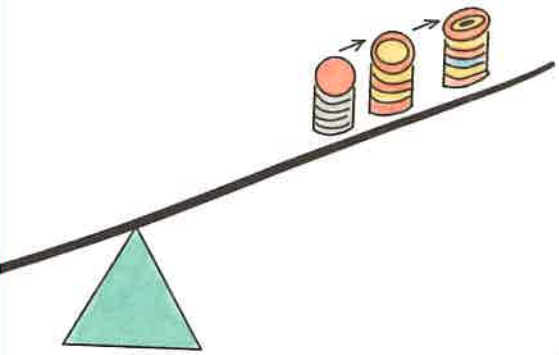


Spolupráce se zákazníkem před vyjednáváním o smlouvě

Proč vlastně vytváříme nějaký produkt? Chceme ho prodat, že? Když stavíme produkty pro zákazníky, je dobrý nápad se jich zeptat, co vlastně chtějí koupit. Jistě, zákazník mnohdy neví, co vlastně chce, a často mění názor, ale dlouhodobou spoluprací můžeme zákazníka „vychovat" a společně vytvořit spolupracující tým, který zajistí jak úspěch náš, tak úspěch zákazníka. Je lepší se se zákazníkem dohodnout a spolupracovat, než se potkávat pouze u soudu a komunikovat přes právní zástupce.

Smlouvy **jsou** důležité, ale neměly by být prostředkem nahrazujícím spolupráci a komunikaci. Ostatně není od věci se už při jejím podpisu zamyslet nad tím, co je opravdu reálné, že se při spolupráci stane – tedy že zákazník možná ne vždy ví přesně, co chce, a že se stejně bude měnit scope – a smlouvu se tak pokusit přiblížit realitě.

Spokojený zákazník, který dostal to, co potřebuje, vás doporučí dalším firmám, zatímco zákazník, který sice má to, co si objednal a ve smlouvě podepsal, ale nijak mu to v jeho misi nepomohlo, se příště podívá po jiném dodavateli. Smlouvy určitě pište, ale ani sebelepší smlouva vás neochrání před změnami, které přijdou. Obvykle to stejně bez ohledu na smlouvu skončí kompromisem, kdy po měsících hádek o chyby a change requesty jich dodavatel polovinu dodělá jako opravu na vlastní náklady a polovinu jako novou funkčnost za extra peníze od zákazníka. Vzájemné spokojenosti a dobrým vztahům to nijak nepomůže.

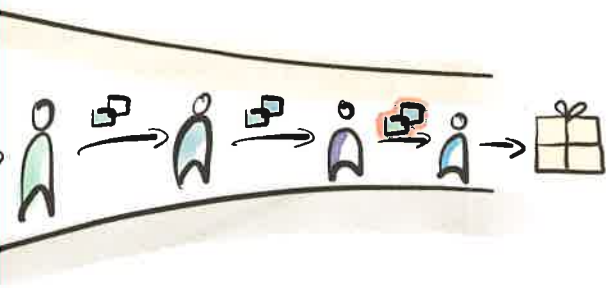


Reagování na změny před dodržováním plánu

Svět se pořád hýbe kupředu a technologie se neustále mění a spolu s nimi se mění i požadavky a problémy našich zákazníků. Ti, jako všichni ostatní, se musí přizpůsobovat trendům na trhu a konkurenci. A my, jako jejich dodavatel, je nemůžeme v těchto změnách brzdit – tedy pokud chceme dále spolupracovat. Představte si, že zákazník přijde se změnou v posledních fázích projektu a ta změna bude důležitá pro jeho přežití na trhu.

Řeknete mu, že se budete držet plánu a kontraktu a změny přijdou až po doručení první verze. A opravdu si myslíte, že další změny budou? Odvážím se tvrdit, že pokud se my dodavatelé nepřizpůsobíme, náš zákazník už si nic dalšího nekoupí, zkrátka proto, že ho konkurence převálcuje. Podobně je to i v méně vyhraněných případech, kdy zákazník prostě jen zjistí, že to, co původně tolik chtěl, vlastně vůbec neřeší jeho problémy a že by naopak potřeboval něco úplně jiného, o čem se předtím vůbec nemluvilo.

Projektové plány **jsou** důležité jako vodítko, ale neměly by řídit životy spolupracujících firem. Každé plány se mění a dogmatické dodržování původních plánů mnohdy vede k větší katastrofě než jejich postupné přizpůsobování dané situaci.



Co je to „lean“?

Lean na druhou stranu je proces převzatý z tovární výroby. Štíhlý. Dělejte věci, jen když jsou potřeba. Just in time. Často se používá i výraz systém tahu. Stejně tak jako agile i lean je spíše o přístupu než striktním procesu. Navíc agile a lean jsou si v mnohém podobné a vzájemně se prolínají. Je to takový pěkný moderní buzzword. Lean firma.

Spousty velkých firem Lean principy implementují, obvykle bez většího porozumění jejich zaměstnanci. Často to končí tím, že si udělají nějaké lístečky a jsou dostatečně lean, tedy štíhlí. Jenže na to, aby to přineslo nějaké výsledky, je stejně jako u agilních metod třeba porozumět filozofii. A ne jen slepě vykonávat nějaké rituály. O co tedy jde? Jednoduše řečeno o omezení práce na tom, co by nemuselo přinášet hodnotu, a tedy v konečném důsledku mohlo přijít nazmar.

Nejznámější lean firma je určitě Toyota. Tam vyvinuli proces řízení výroby odlišný od běžných procesů, kdy vyrábíme kdykoli a cokoli na sklad. Řídí výrobu systémem tahu, kde vyrábíme příslušný díl, až když je opravdu potřeba.

Jak takový princip použít ve firmách, kde žádné fyzické díly nevyrábíme? Tak se třeba podívejme na standardní vývojový proces – waterfall. Nejprve uděláme na sklad analýzu, pak kód a pak testy. A čekáme, že to je tak v pořádku a že všechny díly jsou kvalitní (tedy že design už se nezmění, v kódu se nenajde chyba a že zákazník to tak opravdu chce). A stejně jako ve výrobě se nám děje, že jednotlivé věci musíme předělat, opravit, zahodit. Někdy i celou krabici dílů se stejnou chybou (někdy i celou rozsáhlou funkcionalitu našeho softwaru).

Jak na to? V kostce: omezte rozdělanou práci ('work in progress') a soustřeďte se na to, abyste jednotlivé požadavky dokončili co nejrychleji. Implementujte systém tahu a nezačínejte s analýzou, dokud nemáte prioritní požadavek od zákazníka. A dokud nemáte zpětnou vazbu, že předchozí požadavek byl akceptován.

Aby to bylo celé více uchopitelné, Lean Software Development je založen na následujících principech:

- **Odstraňte vše, co nepřináší hodnotu** – tedy zbavte se odpadu. Pracovat na něčem, co se ve finále vyhodí, je škoda času. Když se vám podaří tento čas investovat do věcí, co mají smysl, budete jistě efektivnější.
- **Zlepšujte se a učte se již v průběhu** – když jen slepě vykonáváte předpisy a sledujete procesy, může se stát, že stejnou chybu opakujete pořád dokola a „odpad“ se vám tedy na konci projektu nahromadí víc, než byste si přáli. Pravidelná zpětná vazba vám pomůže se soustředit jen na to, na čem opravdu záleží.

- **Rozhodujte se co nejpozději** – čím později rozhodnutí padne, tím více máte informací. Takže jsme zase zpět u myšlenky, že nemá smysl vyrábět zásoby na sklad jen proto, že zrovna máte volnou linku nebo programátory.
- **Dodávejte práci, jak nejrychleji to jde** – čím dříve něco dokončíte, tím dříve dostanete zpětnou vazbu, kterou můžete hned v další iteraci zohlednit.
- **Dejte týmu důvěru a zodpovědnost** – a budete mít mnohem motivovanější tým, než když se budete držet tradičních top-down struktur.
- **Zaměřte se na celkový výsledek** – jednotlivé chyby a selhání nejsou podstatné, jestliže se z nich poučíte. „Think big, act small, fail fast; learn rapidly" – tedy Přemýšlejte dopředu, začněte u malých věcí, ty vyhodnoťte a rychle se z nich poučte. Jen tak zajistíte, že výsledný produkt bude úspěšný. Produkt není jen výsledný software, zaměřte se na celkový dojem, který produkt vyvolává. Dbejte na kvalitu a celkovou udržitelnost systému, nevytvářejte technický dluh.

Metoda, která aplikuje na softwarový vývoj myšlenky leanu, se jmenuje Kanban a stojí někde na pomezí agilních metod a leanu. Ostatně obě metodiky staví na stejném filozofickém základu a je často těžké je oddělit. Tato kniha není primárně o lean principech, ale o agilních metodikách. Takže i když se o lean občas otřeme, do detailu se tím dále zabývat nebudeme.

Co je to Scrum?

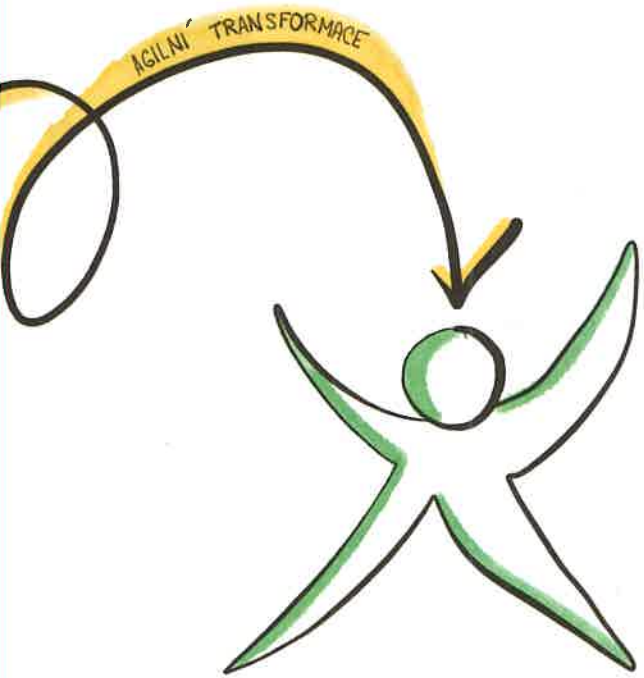
Scrum je v současnosti jedním z nejúspěšnějších a také nejpoužívanějších frameworků, jak se stát agilními. Hodí se na komplexní prostředí, kde je těžké věci naplánovat, a tak je z pohledu businessu třeba výsledný produkt iterovat, flexibilně reagovat na změny, ale vyhnout se chaosu a strategicky ho řídit. Typicky vývoj produktu. Scrum je framework[5] velmi jednoduchý na pochopení, ale jeho aplikace vyžaduje zásadní změnu přístupu, myšlení a mindsetu. Ale o to vlastně jde.

Co je to Kanban?

Kanban je proti Scrumu ještě volnější a to je také jeho hlavní slabinou. Nenutí vás totiž se nijak změnit. Kanban se hodí na prostředí, kde nic strategicky řídit nechcete a chcete jen co nejrychleji reagovat na změny. Typicky call centrum.

Co je to Scaling (škálování)?

Scaling se používá na prostředí, která jsou větší než jeden tým. Case-studies, jak na to, je mnoho (například videa Spotify), frameworků, které jsou univerzálněji popsané, je také nespočet, například Large-Scale Scrum – LeSS, který je asi nejbližší filozofii a hodnotám Scrumu. V závěru této knihy se budeme věnovat zkušenostem se scalingem, tedy škálováním Scrumu na větší celky a agilním transformacím velkých firem.



Agilní transformace

Agilní transformace je náročná změna. Je to změna myšlení a přístupu. Není to jen nový proces, co se někam napíše a pojede se dál. Není to jen o nějakém nástroji nebo nové roli. Je to změna kultury v celé organizaci. Agile je cesta. Nikdy nebudete hotoví. Neustále budete nacházet nové a nové metody, jak zlepšit svůj styl práce, jak najít inovativní řešení, jak se zdokonalit. Nezapomeňte, že „agile" by nikdy neměl být váš cíl. Agile je jen cesta, jak vašich strategických cílů dosáhnout.

Vývoj softwaru

Agile v prostředí vývoje softwaru začal. Čím se vlastně vývoj softwaru liší od standardní továrny? A proč klasické metody řízení projektů právě tady selhávají? Předně je vývoj softwaru komplexní a empirický problém. Nejde jen tak něco naplánovat a podle plánu vytvořit. Problém je v tom, že jakákoliv empirická činnost se velmi špatně odhaduje a ještě hůř se plánuje.

Softwarové společnosti s tím zápasí odjakživa. 70 % projektů končí pozdě, jsou mimo budget a velmi často dodají něco úplně jiného, než zákazník potřeboval. Poté následuje nekonečné dohadování o jednotlivých požadavcích, které ovšem končí vždy stejně – část zaplatí zákazník jako změnu, část dodavatel jako opravu. A o spokojeném zákazníkovi si můžeme nechat zdát.

Prvním problémem je, že zákazník často neví, co chce. Tedy vždy si myslí, že ví naprosto přesně, co chce, ale nedokáže to efektivně předat svému dodavateli. Aplikaci si neumí představit, má jen hrubou představu o tom, jak by mu měla pomoci a urychlit či jinak zlepšit práci. A typický zákazník už vůbec nerozumí počítačům, neví, proč by měl použít knihovnu X nebo technologii Y. Bohužel stále většina softwarových firem své zákazníky nutí říct dopředu přesné požadavky s představou, že oni to pak podle nich jen vyrobí, stejně jako na tovární lince.

Typicky požadavky rozmyslí tým analytiků, popíše, navrhne technologii, architekturu, design, cosi zdokumentuje a předá vývojářům. Ti interpretují specifikaci podle svého a něco nakódují. A když zbude čas, hodí to ještě testerům na otestování. Na závěr to předáme zákazníkovi a o problém je postaráno; s překvapením sledujeme upřímné zděšení zákazníka následované slovy, že očekával něco jiného. Že pracuje jinak, než jsme si mysleli, a že naše geniální funkcionality ani nepochopil. A hlavně – vůbec není spokojený. Přitom my jsme napsali přesně to, co si objednal. Výsledek? V mnoha firmách to skončí tak, že „příště si musíme dát pozor a sepsat ještě detailnější specifikaci, aby se nám to nestalo". A bludný kruh se uzavírá … anebo ne?

Proč vůbec něco měnit

Hned na začátku vás musíme upozornit, že vás čeká spousta nelehké práce. Totiž zavádění agilních metodik, to není jen o změně procesu nebo kolonek na formulářích. Iniciátorem celé změny je obvykle palčivý a zdánlivě neřešitelný problém, který vás pálí natolik, že se vám přechod na agilní metody vyplatí. Každá změna je těžká a změna kultury rozhodně nejtěžší.

Čeká vás změna firmy z hierarchicky řízené na firmu orientovanou na problém. Budete chtít dosáhnout něčeho, čemu se říká samoorganizovaný tým. Najednou vaše práce nebude založená na práci jednotlivců řízených odněkud shora, ale na spolupráci lidí v týmu, na jejich schopnosti se domluvit, rozhodnout a nést za svá rozhodnutí odpovědnost. Pro některé firmy to zní jako utopie, pro jiné je to blízké tomu, jak již dnes fungují. V obou případech se ale změní hodně, a to nejen u vývojářů a testerů.

Jaké jsou nejčastější důvody pro přechod na agilní metody?

Flexibilita – do nedávna to šlo. Dělali jste releasy jednou za 6 až 12 měsíců, zákazníci na to byli zvyklí a firma také. Když chtěli nějakou změnu, protáhli jste ji vaší SW výrobní linkou a při troše štěstí zákazník svou změnu dostal už za několik měsíců. Ale doba se změnila a najednou všichni chtějí všechno hned. Chtějí dostávat výsledky po malých kouscích, ideálně ihned, jak jsou hotové. Bohužel toto v současném procesu není možné. Analytici musí požadavek popsat, vývojáři nakódovat, testeři otestovat a to není jen tak. Navíc nemůžete releasovat kdykoli. Tím byste ohrozili plán releasu a celá snaha by skončila chaosem.

Efektivita – máte pocit, že by toho šlo stihnout víc, kdybychom šli všichni za jedním cílem? Studie ukazují, že spolupráce více lidí je výrazně efektivnější než práce jednotlivců. Co se frameworků týče, máte v podstatě dvě možnosti. Zkusit párové programování, kde máte na všechny činnosti dva lidi, tedy i dva SW vývojáře u jednoho počítače. Zkušenosti ukazují, že tato metodika je efektivnější, kvalitnější a rychlejší, než když každý z nich pracuje samostatně. Anebo postavit spolupracující tým, jak doporučuje Scrum proces, kdy jednotliví členové spolupracují na jednom výsledku, pomáhají si a sami se organizují. Chvíli trvá, než si tým na sebe zvykne, ale funguje to skvěle.

Ještě na jeden aspekt se můžete zaměřit. Podívejte se na to, jak řídíte funkcionalitu. Kdo je zodpovědný za produkt a kdo za konkrétní funkčnost. Dělejte jenom na tom, co opravdu je potřeba. Na tom, co zákazník potřebuje a použije. Na tom, co mu přinese v daném čase co nejvyšší hodnotu. Systémy jsou plné funkcionalit „co kdyby". Budete-li dobře řídit funkcionalitu, a i v tom vám agilní metody pomůžou, můžete ušetřit až 80 % času.

telnost – jestli vaše projekty končí včas sčasů před koncem relasu, asi vám váš pro- je dobře. Pokud to tak není, agilní metody ou pomoci i s tímto aspektem. Tyto metody zcela odlišný způsob odhadování. Odhadují ch jednotkách a do odhadování zapojují celý u to zní hodně zvláštně, ale přesnost vašich se časem výrazně zlepší. Rozdělit projekt ousky je druhou důležitou metodou, jak zlep- datelnost. Na krátkém úseku se méně proje- studentský syndrom a effort, který tým inves- bilnější, bez výkyvů a stresu na konci.

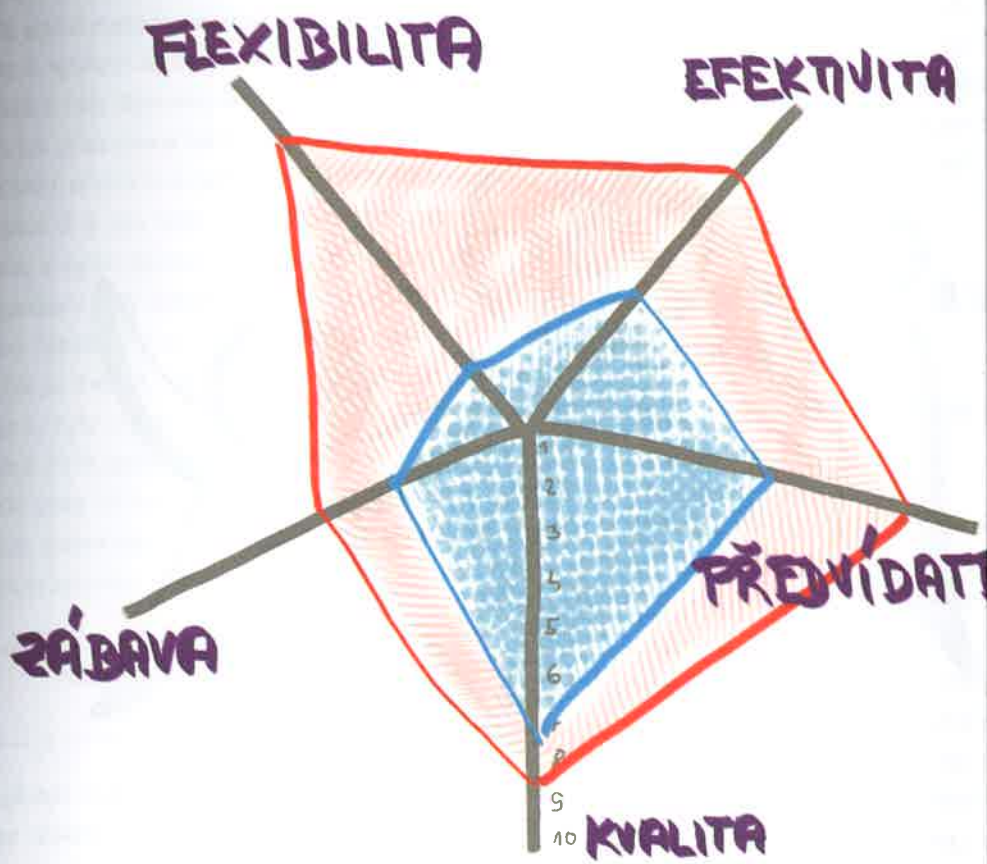
tady cílíme na dvě oblasti. Za prvé zapojíme ktu zákazníka. Zeptáme se ho, co chce, proč, a jak to bude používat. Ukazujeme mu výsledky kouskách a tím řídíme jeho očekávání. Nestává by zákazník produkt odmítl jako nepoužitelný jeho přepsání. Na druhé straně tím, že uděláte zodpovědný za kvalitu výsledku, se sníží počet zroste dlouhodobá udržitelnost kódu. Tým se o to, že jim neroste technický dluh. Pro oba vádějí agilní metody spoustu praktik.

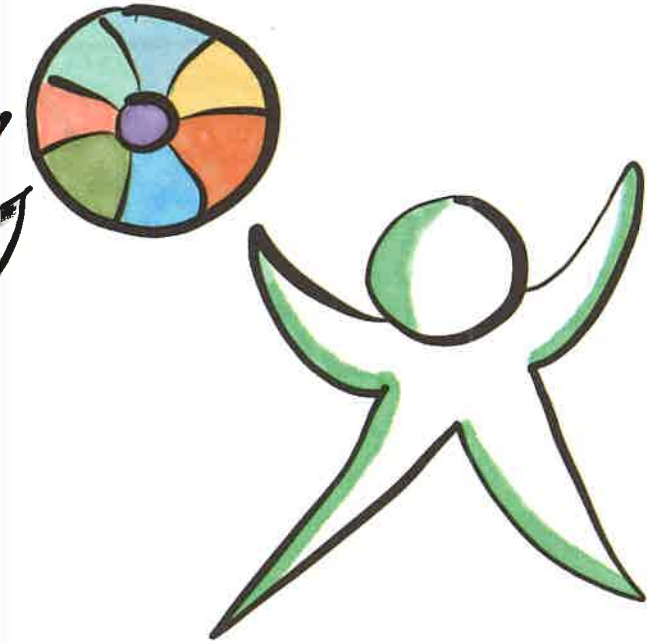
a v neposlední řadě, práce bude zase zábava. ví členové budou vědět, co a proč píšou. nápat smysl produktu a rozumět zákazníkovi. olupráce s ostatními nakonec také přináší motivovaný člen týmu je jistě efektivnější než vývojář sedící sám za svým počítačem.

*Než se pustíte do nasazování agilních metod, udělejte si jasno v očekáváních, zejména týkajících se **flexibility**, **efektivity**, **předvídatelnosti**, **kvality** a **zábavy**.*

Jak jsme již říkali, nic není zadarmo. Jestli jste v některé kategorii našli dostatečně velký potenciál, pusťte se dál do čtení. Jestli ne, možná, že vám současné projekty fungují dobře a na změnu ještě nenastal čas. Nasazovat agilní metody jen proto, že je to něco nového, nemá smysl. Je to náročný a trnitý proces, na jehož konci vás musí čekat dostatečně velká odměna. Jinak to vzdáte ještě dřív, než se změny stihnou projevit.

Zkusit si to můžete na jednoduchém cvičení. Výše zmíněné důvody pro změnu si nakreslete do grafu a ohodnoťte jednotlivé kategorie na stupnici 1–10, kde 1 je špatné, 10 je super. Hodnotíte váš tým, projekt, produkt, firmu. Ideální je tzv. pavučinu nakreslit ve dvou barvách. První odpovídá realitě, tedy současnému stavu, a druhá odráží vaše očekávání za, řekněme, rok. Oblasti, ve kterých je největší rozdíl hodnot, indikují důvody pro změnu. Jste-li naopak se vším spokojeni, pak nejlepší strategií bude zůstat u současných procesů a nic neměnit.





Business Agility

V poslední době se mimo IT obor často skloňuje termín Business Agility. Co si tedy pod Business Agility představit? Agilní metody v IT začínaly. Ale doba se posunula, agilní metody se osvědčily, a tak více a více slyšíme o aplikaci Agile mimo IT, o zapojeni businessu. V kostce tedy Business Agility. Organizace hledají způsob, jak přistupovat ke svému businessu tak, aby byly flexibilní, efektivní a rychleji reagovaly na změny. Agile v rámci IT je jen částí úspěchu. Aktivní zapojení businessu a agilní strategické řízení firmy je důležité, aby organizace byly schopny efektivně přežít z dlouhodobého hlediska v prostředí, které je obecně globálně rychle se měnící, turbulentní a není tak předvídatelné, jako to bylo v minulosti. Nevěříte? Malá statistika: přibližně 70 % společností, které byly v žebříčku Fortune 1000 před 10 lety, z něj zmizelo, nestačily se přizpůsobit změněnému prostředí – digitalizace, všudepřítomný internet, mobilita.

Adaptabilita, flexibilita a vyváženost jsou tři klíčové elementy Business Agility.

Business Agility je organizační změna, která funguje bez ohledu na podstatu businessu, kde organizace působí. Business Agility je vlastně agilita z pohledu businessu, je to změna kultury a fungování celé organizace, její schopnost se adaptovat, zregenerovat, rychle se změnit na základě vnitřních, ale hlavně vnějších změn. Řízení se stane daleko dynamičtější, než jsme zvyklí z prostředí ročních plánů a budgetů, iterace businessu se točí v krátkých cyklech, přináší inovace a evoluce. Můžete namítnout, že to není nic nového. No není. Je to jen ten stejný agile přefrázovaný do jiných slov, která jsou více uchopitelná businessem a managementem, a má tím pádem potenciál transformovat firmu z úplně jiné strany.

Agilní organizace a agilní leadership

Ještě o krok dál, než jde Business Agility jde koncept agilní organizace.

V současnosti, kdy je většina problémů, se kterými se organizace potýkají, komplexní, je třeba umět na změny reagovat. Být dostatečně flexibilní, agilní, chcete-li. Moderní organizace jsou uzpůsobené tak, aby neustále hledaly a optimalizovaly business hodnotu. Už není tak důležité, jak efektivně pracují jednotlivci. Podstatný je celkový výsledek a přínos. Komplexita moderního světa s sebou přinesla takovou složitost, že již není možné věci rozmyslet, popsat procesy, vytvořit zodpovědné role.

Organizace potřebují tvořit adaptivní sítě, které jsou inovativní a kreativní a rychle reagují na změnu prostředí.

Odpovědnost se z jednotlivce přenáší na týmy.

Současně s tím je třeba změnit leadership model, kde týmy samy přebírají zodpovědnost a rozhodování v rámci firemní strategie do svých rukou a pomáhají organizaci stát se flexibilní. Není to žádný chaos, taková firma má na rozdíl od mnoha klasických firem velmi jasně definovaný směr. Funguje jako flotila malých lodí, které všechny plují za jedním cílem, ale mají v rámci své mise autonomní řízení.

Organizace je tak dobrá, jak dobré má leadery.
Změňte svůj styl vedení a organizace vás bude následovat.
Decentralizujte, tvořte sítě and komunity.
Umožněte autonomní rozhodování v rámci dobře definovaného prostoru.

A na závěr: co je opakem agilní firmy? Firma byrokratická, hierarchická, procesní. Prostě firma s klasickým fixním řízením, pevnými strukturami, kde každá změna musí být schválena na vyšších úrovních a autonomie neexistuje.

Část II
Popis metod, p
praktik a art

Ve druhé části popíšeme jednotlivé artefakty, procesy a techniky. A protože jedno souvisí s druhým, na úvod dáváme krátký slovníček agilních pojmů. Detailní vysvětlení, jak konkrétní praktiku implementovat, co od ní očekávat, jak začít a čeho se naopak vyvarovat, najdete v následuj čeními, jak souvislost kapitolou aby daná

Agilní slovníček

Praktiky, procesy a artefakty	Vysvětlení
Backlog Refinement / Backlog Grooming	Pravidelná aktivita, na které tým probírá s Product Ownerem položky z Backlogu tak, aby porozuměl celkové vizi, dodávané business hodnotě a byl schopen naplánovat další Sprint. Někdy může být formou meetigu, většinou ale probíhá distribuovaně.
Definition of Done	Domluvená pravidla pro předání, definující hotovou funkcionalitu.
Epic	Větší funkční celek, který se následně rozpadá na menší User Stories.
Extreme Programming – XP	Metoda agilní spolupráce postavená na spolupráci a programovacích praktikách jako review, continuous integration, Test Driven Development (TDD), pair programming, …
INVEST	Kritéria dobře napsané User Story: Independent, Negotiable, Valuable, Estimable, Small, Testable.
Kanban	Metodika na pomezí agile a lean fungující na principech vizualizace procesu, omezení rozpracované práce a minimalizace času potřebného na dokončení jednotlivých celků.
Multifunkční tým (cross-functional tým)	Agilní metody přichází s konceptem multifunkčních týmů, které nahrazují původní komponentní týmy, které byly zaměřené na konkrétní komponentu nebo technologii. Multifunkční týmy mají všechny znalosti potřebné k tomu, aby dokázaly vzít libovolnou položku s Product Backlogu a tu dokončit. Tedy jako tým dodat jakoukoli ucelenou funkcionalitu, která přináší hodnotu zákazníkovi, a to napříč systémy i technologiemi.
Planning	Ve Scrum procesu plánuje tým, vybírá z prioritních User Stories do Sprint Backlogu ty, které je schopen dokončit v příštím Sprintu..
Planning Poker	Metoda pro odhadování velikosti User Stories v relativních jednotkách, tzv. bodech. Obvykle se hraje ve formě herních karet s tzv. Fibonacciho řadou pro určení velikosti (komplexnosti) User Story.

Praktiky, procesy a artefakty
Položka Backlogu
Product Backlog
Product Owner
Retrospektiva
Review meeting
Rychlost (Velocity)
Scaling (škálování)
Scrum
Scrum Master
Scrum Tabule

Praktiky, procesy a artefakty	Vysvětlení
Položka Backlogu	Definuje funkcionalitu která přináší hodnotu. Často je zapsaná formou User Story.
Product Backlog	Prioritizovaný seznam funkcionality (user stories), kterou chceme dodat zákazníkovi. Backlog je priorotizovaný Product Ownerem. Na jeho vzniku se podílí celý tým a často i stakeholdeři a zákazníci. Je vždy přístupný celému týmu.
Product Owner	Vlastník produktu. Má na starosti vizi, Product Backlog a je zodpovědný za ROI a celkovou úspěšnost Produktu. Je součástí týmu. Definuje, co je potřeba v daném produktu či oblasti udělat. Na základě kontaktu a diskuzí se zákazníky určuje prioritu úkolů.
Retrospektiva	Týmový meeting, kde jednotliví členové zhodnotí, co se jim při poslední iteraci dařilo, v čem chtějí pokračovat a co naopak chtějí vylepšit či změnit.
Review meeting	Po skončení každého Sprintu tým v rámci Review prezentuje zákazníkovi funkční inkrement produktu s cílem získat zpětnou vazbu.
Rychlost (Velocity)	Rychlost měříme za tým a Sprint v relativních jednotkách, tzv. bodech. Slouží k lepšímu odhadu práce pro příští Sprinty a je též ukazatelem efektivity týmu. Rychlost má smysl pouze v daném týmu, a jelikož používáme relativní jednotky, nelze rychlost vzájemně porovnávat mezi týmy.
Scaling (škálování)	Scaling se používá při aplikaci agilních frameworků na prostředí více týmů.
Scrum	Proces postavený na týmové spolupráci, zapojení zákazníka a pravidelné zpětné vazbě v krátkých Sprintech. Scrum proces je v současnosti jedna z nejúspěšnějších a nejčastěji využívaných agilních metodik.
Scrum Master	Coach a facilitátor týmu. Odstraňuje překážky, stará se o rozvoj a fungování týmu. Udržuje Scrum proces v chodu. Není manažerem týmu.
Scrum Tabule	Zobrazuje stav aktuálního Sprintu. Obvykle pomocí papírových lístečků s jednotlivými User Stories a tasky.

Praktiky, procesy a artefakty	Vysvětlení
Self-organized tým	Tým, který se v rámci mantinelů projektu organizuje a sám rozhoduje, na čem a jakým způsobem bude pracovat, bez vnějšího vlivu tak, aby maximalizoval efektivitu, flexibilitu a motivaci jeho členů. Pozor, nezaměňovat s anarchií, i v self-organized týmech vždy platí, že změnit lze pouze něco = existují hranice a procesy, které ani tento tým nemůže změnit.
Sprint	Krátká fixně dlouhá iterace, ve které se tým zaváže dodat funkcionalitu, která má pro zákazníka hodnotu.
Sprint Backlog	Vybraná funkcionalita z Product Backlogu pro daný Sprint.
Sprint Goal / Cíl Sprintu	Sprint Goal, tedy cíl Sprintu, je to taková malá vize na jeden Sprint. Správný Sprint Goal adresuje potřeby zákazníků, je zaměřený na hodnotu, nikoli na funkcionalitu. Dodává Sprintu smysl a hodnotu.
Standup / Scrum Meeting	Krátký týmový meeting, pravidelně každý den, obvykle ráno. Jednotliví členové sdílí mezi sebou aktuální stav a identifikují případné problémy. Facilitován Scrum Masterem.
Story point (Bod)	Relativní jednotka velikosti user story. Nedá se převádět na čas a má smysl pouze v jednom daném týmu, tedy nemůžeme porovnávat Story Pointy mezi jednotlivými týmy.
User Story	Funkcionalita popsaná v tzv. kanonické formě: Jako Uživatel chci Funkcionalitu, abych dostal Business Value. User Story je jednou z možných forem zápisu Položek Backlogu, měla by být dostatečně malá, aby jí tým rozuměl a mohl ji naplánovat na další Sprint.
Zákazník	V agilním světě je pojem zákazník poměrně široký a zahrnuje všechny, co od produktu něco očekávají jak zevnitř, tak i zvenku firmy, jako např. stakeholder, manažer, marketing oddělení, obchod, podpora, uživatelé.

Obecné pojmy:

Výraz
Budget
Demo
Checklist
Change request
KPI – Key Performance Indicator
Stakeholder
Waterfall

Obecné pojmy:

Výraz	Vysvětlení
Budget	Rozpočet projektu.
Demo	Předvedení produktu, ať již interně, nebo zákazníkům.
Checklist	Seznam bodů se „zaškrtávátky". Nejčastěji slouží jako postupný seznam úkolů či náležitostí nějaké činnosti, které můžeme zkontrolovat a odškrtnout.
Change request	Požadavek na změnu oproti původnímu zadání.
KPI – Key Performance Indicator	Klíčový ukazatel výkonnosti. Jedná se o stanovenou či měřenou hodnotu, proti které porovnáváme skutečný stav efektivity nebo výkonu organizace.
Stakeholder	Stakeholder je zástupce jisté zájmové skupiny zákazníků.
Waterfall	Tradiční metodika vývoje softwaru a jiných projektů, spočívající v řetězení činností za sebe, tzn. dokud se nedokončí předchozí fáze, další nemůže začít.

Lidi a vztahy

Scrum Master

Scrum je založen na principu self-organized týmu, transparentní komunikaci a otevřené kultuře, která podporuje spolupráci, a sdílení informací. Aby to celé fungovalo, zavádí některé specifické role, které tradiční metody managementu neměly – a to Scrum Mastera a Product Ownera.

Scrum Master je koučem, facilitátorem a takzvaným servant leaderem, je to role hodně odlišná od klasického teamleadera. Jeho hlavním cílem je vytvořit samostatný, efektivní a spokojený self-organized tým. Detailnější cíle a povinnosti by se daly shrnout do následujících bodů:

- ☐ pomáhá týmu aby dobře fungoval, stal se takzvaně high-performing a dosahoval tak lépe svých cílů
- ☐ dobrou facilitací pomáhá týmu odstraňovat problémy
- ☐ motivuje tým k lepším výsledkům
- ☐ koučuje tým a stará se o jeho rozvoj

Dále se stará se o to, aby Scrum proces byl efektivní a fungoval, má na starosti jeho dodržování, ale zároveň i možnost iniciovat změnu, je-li to třeba. Scrum Master by měl upřednostňovat koučovací principy, podporovat tým a jednotlivce v jejich rozvoji, být komunikativní, vnímavý a utlumovat případné konflikty v rámci týmu. Scrum Master by tedy neměl být direktivní.

Scrum Master je ten, kdo „zametá cestičku", aby se týmu dobře šlo a pracovalo. Když dobře fungující tým přirovnáme ke stroji, stará se, aby se „kola týmu točila" jak nejrychleji to jde, a nikde to nedrhlo, nevrzalo, neskřípalo. Scrum Master je součástí týmu a měl by být týmu kdykoli k dispozici. Měl by proto sedět v jedné místnosti s týmem.

Funguje-li Scrum Master dobře, nedá se jeho pozice chápat jako náklad navíc. Tím, že rozpohybuje tým a zajistí tak vyšší efektivitu, kvalitu a motivaci, ušetří v celkovém rozsahu více peněz, než Scrum Master jako nákladová položka firmu stojí.

Není proto vhodné roli Scrum Mastera kombinovat s Product Ownerem, který má z definice své role často konfliktní cíle. Obvykle nefunguje ani kombinace role s vývojářem či testerem, kde vývojáři či testeři často upřednostní svou technickou prací před prací Scrum Mastera, a nechají tak tým v kritických okamžicích de facto bez Scrum Mastera.

Technicky nejzkušenější člen týmu obvykle není ideálním Scrum Masterem, protože upřednostňuje své

znalosti a zkušenosti před koučováním, facilitací a rozvojem ostatních. Je to člověk, který dokáže poradit a zná obvykle odpověď na otázky lépe než jiní členové. Scrum Master musí být vnímavý, klást otázky a podporovat schopnost týmu si na řešení problémů přijít sám i za cenu lokální neefektivity.

Když už uvažujete o kombinaci s nějakou jinou rolí, snažte se, aby role Scrum Mastera byla vždy prioritní. Některé firmy nechávají Scrum Mastera koučovat dva týmy najednou, ale ani to není ideálním řešením, protože problémy často chodí spolu. Velmi tedy záleží na prostředí a týmu a konkrétní osobnosti Scrum Mastera. Ani kombinaci rolí Scrum Mastera a manažera rozhodně nemůžeme doporučit. Scrum Master by tým neměl nijak přímo řídit ani organizovat, ani za tým rozhodovat, protože pak se mu bude jen těžko budovat self-organizad tým.

Scrum Master musí být empatický, komunikativní, být dobrý facilitátor, kouč a v neposlední řadě agilní nadšenec.

Technicky nejzkušenější člen týmu obvykle není dobrým Scrum Masterem.

Role Scrum Mastera je pro úspěch produktu klíčová, vyžaduje vnímavého člověka se schopností dobře komunikovat, koučovat, facilitovat. Prostě někoho, kdo je schopen tým posunout dál a udělat jeho členy ještě úspěšnější. Navíc dobrý Scrum Master problémům předchází, tedy se po čase může zdát, že tým už je plně funkční sám a Scrum Mastera nepotřebuje. Když ovšem takového Scrum Mastera přesunete na jinou práci, po čase se v týmu začnou množit problémy a přestane fungovat. Není to hned, o to méně si na konci vzpomenete, že by to mohlo mít souvislost s neexistujícím Scrum Masterem.

Výborný Scrum Master by měl být schopen pracovat na třech různých úrovních #ScrumMasterWay[3] konceptu – „Můj tým", kde se věnuje primárně rozvoji self-organized development týmu, „Vztahy", kde se věnuje vazbám na okolí, vztahům s ostatními týmy, zákazníky, manažery, či product ownerem a vytváří z nich fungující self-organized ekosystém, a „Celý systém", kde pomáhá celé organizaci agilní přístupy aplikovat a stát se tak agilní organizací.

Říkali jsme, že Scrum Master nemá být direktivní, není to manažer týmu. Na druhou stranu, nemůže být ani nevýrazný a nechat vše plynout. Role Scrum Mastera se mění v závislosti na zkušenostech a schopnostech týmu se sám organizovat.

Ze začátku může Scrum Master více radit a určovat co se bude dít, více všechno organizuje a svým způsobem řídí. Jste-li ale v takovéto roli i po několika měsících, pravděpodobně jste zapomněli, že cílem Scrum Mastera je vybudovat self-organized tým, který si bude

schopen sám najít řešení svých problémů a sám se organizovat tak, aby jeho práce byla co nejvíce efektivní. Nebuďte jako starostlivá maminka, která ani v 10 letech dítě nepustí samotné přes ulici na nákup. Nechte týmu prostor, nechte z nich vyrůst sebevědomé a zodpovědné jedince, pravý Scrum tým.

Každá kombinace rolí vždy přináší rizika a tak když už jste si jisti, že je nutné role kombinovat, dejte si pozor, že rozumíte dobře konfliktů cílů a priorit těchto rolí.

Souvislosti:

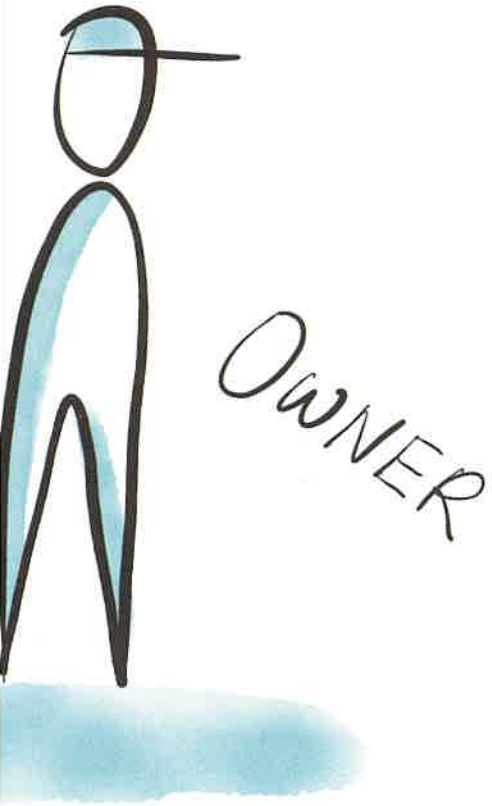
(co budete ještě potřebovat zavést)

Self-organized tým: cílem každého Scrum Mastera je dosáhnout toho, že se tým sám organizuje, rozhoduje a nese za svá rozhodnutí odpovědnost.

Product Owner: jestliže Scrum Master se stará o správně fungující tým, bez Product Ownera, který udává týmu směr a priority, nikdy nedosáhnete správného výsledku.

Development tým: když už jsme u Scrumu, Development tým je ten, který se stará o to, že máme každý Sprint funkční inkrement produktu, na který můžeme od zákazníka získat zpětnou vazbu.

Role Manažera: s
operativě, alokov
vých lidí. To má na
za vytvoření pros
pracovat a podílí s
než na řešení kaž
tým vyřešit spolu
organizacích je vi
gementu k agilním



Product Owner

Další z rolí, které Scrum proces zavádí, je role Product Ownera. Product Owner je vlastníkem produktu. Má na starosti definování produktové vize a její transparentní komunikaci týmu, zákazníkům, firmě. Product Owner definuje priority, rozhoduje, na které funkcionalitě se bude pracovat dříve, na které později a na které vůbec. Má na starosti business hodnotu (business value) a také ROI (navratnost investice) celého produktu.

Product Owner je tedy zodpovědný za celý Product Backlog. Není na takovou práci sám, ale pomáhá mu celý development tým, ale také stakeholdeři, zástupci zákazníka, uživatelů, SW architekti a User Experience specialisté.

Product Owner je Development týmu pravidelně podle potřeby k dispozici, ale na rozdíl od Scrum Mastera už s týmem ne vždy musí sedět v jedné místnosti. Tráví dostatek času se zákazníky, aby vstřebal jejich prostředí a dokázal se vždy správně rozhodnout, kde je pravá hodnota pro zákazníka. Product Owner neřídí ani jednotlivé členy týmu, ani tým, nemá možnost jim přikazovat, co musí dokončit, jen stanovuje, co se má dělat a v jakém pořadí – tedy definuje priority.

Primárním cílem Product Ownera je mít úspěšný produkt. Definuje vizi, hodnotu produktu a priority tak, aby všichni věděli, čeho chceme dosáhnout. A to jak v rámci týmu, tak managementu a zákazníků. Cíl musí být pro všechny stejný, je to to, co vás spojuje a dává všem energii.

Role Product Ownera by neměla být kombinována s rolí Scrum Mastera. Role je pro úspěch celého procesu klíčová, vyžaduje komunikativního člověka se silnými znalostmi produktu.

Product Owner musí vhodně vyvážit obě funkce své role – znát zákazníka a být s ním v kontaktu a zároveň být týmu k dispozici.

Obvykle se čas Product Ownera dělí tak, že je ***80 % u zákazníka a 20 % v týmu****.*

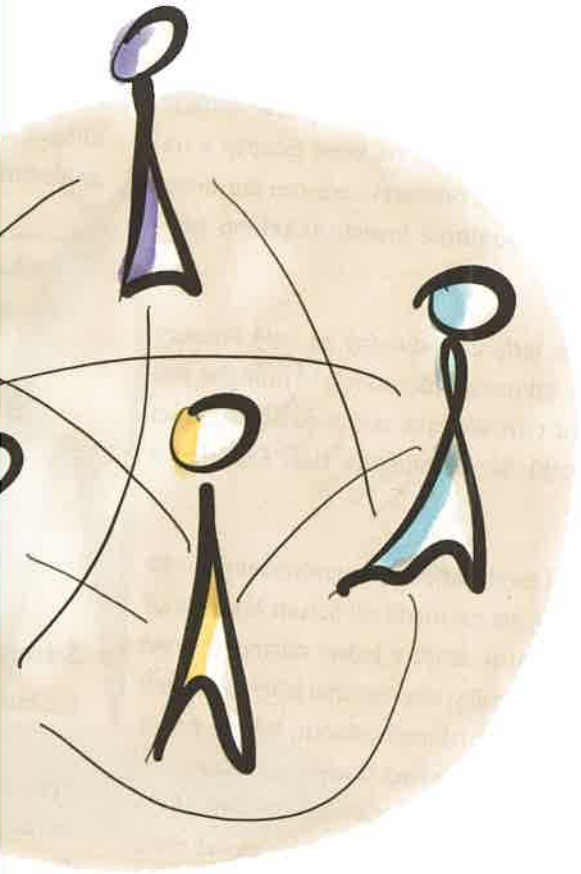
Konkrétní čísla se ale můžou lišit v závislosti na typu produktu.

Souvislosti:

(co budete ještě potřebovat zavést)

Tým: Spolupráce v týmu je lepší než práce jednotlivců. Správný tým zná zákazníka a je partnerem Product Ownera při tvorbě Product Backlogu.

Product Backlog: Product Owner by měl mít jasno v tom, na čem tým bude pracovat, být schopen určit priority.



Self-organized tým

Agilní metody jsou postavené nejen na časté zpětné vazbě a komunikaci, ale i na týmové spolupráci. A vývoj softwaru je komplexní problém, který není snadné naplánovat dopředu. Každá firma je jiná, každý produkt je jiný, každý tým je jiný. Nikdo zvenku neumí dosáhnout optimální efektivity ani workflow. Proto agilní týmy často spoléhají na své jednotlivé členy a nastavují adaptivní proces, který jeho členové sami můžou ovlivnit a změnit.

Nutnou podmínkou pro vznik takového self-organized týmu je mít společný cíl. Tým musí rozumět zákazníkovi, chápat jeho prostředí, vědět, jak bude produkt používat. Jít všichni za stejným cílem, mít stejnou vizi. Dalším stavebním kamenem je důvěra. A to nejen mezi jeho členy, ale i k zákazníkovi a zbytku firmy. Pak už zbývá jen dát týmu možnost se rozhodovat a měnit to, jak pracují. Umožnit týmu podílet se na tvorbě Product Backlogu a přispět tak osobně k tomu že zákazník dostane to nejlepší možné.

Tým musí táhnout za jeden provaz. Selže-li jeden člen týmu, není to jeho problém, ale znamená to, že selhal celý tým. Jestli tým např. nestihl dodat všechny položky ve Sprint Backlogu, protože mají jen jednoho testera a ten ani nemohl za poslední dva dny Sprintu stihnout vše otestovat, není to chyba tohoto testera, ale celého týmu, který se měl zorganizovat tak, aby vše dokončili včas.

Teď asi rozumíte tomu, jak by měl takový self-organized tým vypadat. Občas se ale setkáváme s tím, že to v nějaké firmě přeženou a tým potom má z pozice samoorganizujícího se týmu pocit, že by měl rozhodovat o všem, co se ve firmě děje. Takže pojďme se na takový tým podívat z druhé strany a pojďme připomenout, o čem tým nerozhoduje. Tým nemá možnost se rozhodnout, jestli bude, nebo nebude pracovat podle Scrum procesu a agilních přístupů, ani rozhodovat, kdo bude členem týmu. To se obvykle rozhoduje na úrovni organizace.

Správný tým musí spolupracovat a takzvaně táhnout za jeden provaz.

Selže-li jeden člen týmu, selhal celý tým bez hledání viníka.

Tým nemá tedy ani pravomoc zrušit agilní praktiky či Scrum meetingy, jako např. Standup. Tým se organizuje čistě v rámci daného hřiště a daných pravidel agilního či Scrum procesu. Tým také nerozhoduje o tom, co se bude nebo nebude implementovat, může si pouze vybrat z priorit daných Product Ownerem. Jako obvykle, hledáme nějaký vyvážený stav. Univerzální pravidla zde ovšem nepomůžou, správnou úroveň volnosti si musíte najít sami podle vyspělosti vaší organizace a týmů.

Multifunkční tým

Správný Scrum tým je nejen self-organized, ale i multifunkční a vzájemně zastupitelný. To, že máte v týmu experta na konkrétní systém, komponentu, GUI, nebo databáze neznamená, že se tito experti musí sto procent svého času věnovat jen danému systému, komponentě, GUI či databázím. Tým se sám dohodne, kdo na čem bude pracovat, kdo komu pomůže a kterou znalost by si měl začít pozvolna budovat, aby byl jako celek co nejefektivnější.

Nejde tady o žádnou krátkodobou efektivitu, ale o investici do flexibility týmu. Ze zkušeností na mnoha i velmi komplexních projektech můžeme říct, že to nikdy není tak náročné, jak se vám zdá. Stačí si jen zvyknout na jiný styl práce a vybudovat sdílenou znalost tak na 10 až 20 procent. I to tým výrazně zefektivní a udělá flexibilnější.

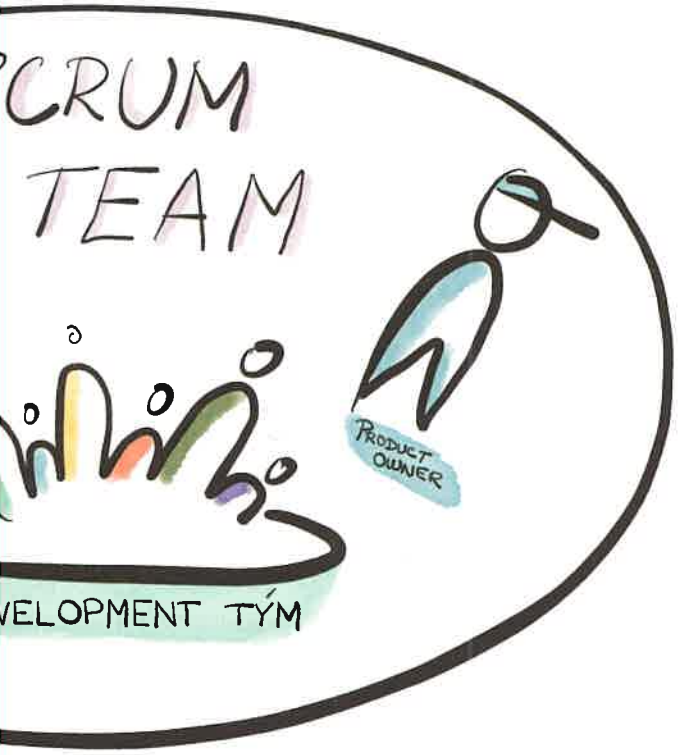
Stejné je to i s rolí testera, vývojáře a analytika. Tester bude pravděpodobně vymýšlet test-casy, bude garantem za testování v týmu. Ale zdaleka to nemusí být ten jediný, kdo v případě potřeby testuje úplně všechno. Všichni, včetně vývojářů, mu mohou pomoci a ve volných chvílích může zas tester pomáhat například s analýzou a specifikací.

Hledání chyb také nemusí končit jen zalogováním nalezených chyb do systému, ale může pokračovat i analýzou, která vývojářům ušetří při opravách čas. Je jen na týmu, jak se jeho jednotliví členové domluví, že budou pracovat. Nejde o optimální vytížení jednotlivce, ale o optimální nastavení celého systému a jeho flexibilitu při reakci na změny.

Další změnou ve spolupráci bude nejen výše popsaná zastupitelnost týmu, ale i jiný styl práce na úlohách. Týmy často ze zvyku nastaví takový malý Waterfall v rámci Sprintu, kdy začínají analýzou, když je hotová, vývojáři to nakódují a následně testeři otestují. Jedním z průvodních jevů takového systému je, že testeři nemají na začátku co dělat a na konci mají naopak šíleně práce. Částečně tomu lze odpomoct tím, že budou spolupracovat i nad rámec klasických rolí, jak bylo popsáno v předchozím odstavci, ale podstata změny je jinde.

Správný Scrum tým na každé položce backlogu spolupracuje již od začátku. Analytik se zamýšlí, jak to bude fungovat a odpovídá na otázky vývojáře, jak se daná funkcionalita má chovat. Oba na ní pracují současně. A s nimi je ve trojici ještě tester, který se na ni dívá z pohledu toho, jak by se mohla daná funkcionalita rozbít, a rovnou identifikuje případné chybové scénáře.

Výhodou takového přístupu je, že všichni pracují na stejné věci, a tedy nenastává žádné přepínání kontextu, kdy se vývojář ptá, jak to analytik myslel, ani nevzniká pocit, že to tester pořád rozbíjí, když už to



přece bylo několikrát hotové. Je to zásadní změna ve stylu práce týmu, ale bez ní nebude váš Scrum tým nikdy úspěšný.

Na závěr, Development tým by měl být alokován na full time a členové týmu by měli sedět v jedné místnosti, aby si mohli dobře sdílet znalosti, zkušenosti, pomáhat si a spolupracovat. Výhodou je, že ubude složité alokace na měsíce dopředu a vše se výrazně zjednoduší. Ale i tohle bývá obzvláště ve velkých firmách ze začátku problém.

Je to o zvyku. Byli jsme roky zvyklí vytvářet superspecializované jedince, kteří pracují samostatně jako jednotlivci, a ty synchronizuje manažer. S agilními přístupy se to otáčí a tým se o alokaci svých členů na jednotlivé úlohy domlouvá sám, spolupracuje, buduje společné know-how. Veškeré úlohy, které šly dříve na specialisty, jdou nyní na tým jako celek a tým už se sám rozhodne, jak s nimi naloží.

Souvislosti:

(co budete ještě potřebovat zavést)

Retrospektiva: Je jedním ze základních prostředků, jak týmu umožnit získat sám na sebe zpětnou vazbu a ovlivňovat proces, to jak pracujeme.

Scrum Master: Jednou z povinností Scrum Mastera je docílit takové spolupráce a komunikace, aby vznikl dobře fungující tým.

Scrum tým a Development tým

Další z rolí, které Scrum proces zavádí, je role Development týmu, který je jak self-organized, tak multifunkční, a je tak schopen dodat zákazníkovi hodnotu na konci každého Sprintu. V rámci Development týmu nejsou žádné role, jeho členové jsou prostě jen členy týmu, kteří společně dělají maximum, aby na konci Sprintu měli funkční inkrement produktu.

Scrum tým potom tvoří tři zmíněné role – Scrum Master, Product Owner a Development tým. Není mezi nimi žádná hierarchie, je to tým.

Zákazník

Agilní procesy jsou jiné i v tom, že se snaží zapojit zákazníka do projektu, aby si sám určoval, jaké jsou jeho priority, a podílel se již v průběhu projektu na jeho změnách a funkcionalitě. Aby se stal součástí týmu. Zákazníkem v tomto kontextu rozumíme kohokoliv, kdo má na projektu nějaký zájem. Tedy to může být jak člověk zevnitř, tak i zvenku firmy, např. stakeholder, manažer, marketing oddělení, obchod, podpora, uživatelé.

V ideálním případě budete chtít zákazníka udělat součástí týmu. Být partnery. Na to potřebujete mít splněno několik aspektů. Znát sebe, své schopnosti a možnosti; znát zákazníka, rozumět jeho potřebám a businessu; a mít vzájemný respekt a důvěru.

Je to dlouhodobá práce. Důvěru musíte budovat postupně. Jednou z možností je pustit zákazníka přímo do Backlogu, ukázat mu jednotlivé User Stories, aby viděl, jak tým pracuje. Už jen to, že ve velmi krátkých pravidelných iteracích ukazujete dokončenou funkcionalitu, je pro zákazníka velmi pozitivní. Vidí, že se něco děje, zná konkrétní členy týmu. Jsou to lidi, kteří mu dodali to, co potřeboval, a když se jim zrovna nedaří, je zákazník obvykle více nakloněn chápat to, že tým narazil na technické problémy, než když v klasickém projektu těsně před koncem zjistí, že vlastně nic není hotovo.

Základem je transparentně komunikovat o úspěchu i problémech, respektovat zákazníka a snažit se splnit více než to, co chce – dát mu to, co opravdu potřebuje. V agilním světě to není nikdy čistý vztah dodavatel – zákazník. Když si vzpomenete na Agilní manifest a princip „Upřednostňujeme funkční software před jednáním o smlouvě" – tohle je jeho konkrétní naplnění. Než se slepě snažit dát zákazníkovi to, co si objednal, udělejte z něj partnera a společně produkt během domluveného času vymyslete a dodejte.

Spolupráce se zákazníkem tedy nemusí být obvyklý fixed time, fixed price model. Jste-li se zákazníkem partneři, obvykle máte domluvenou pouze rámcovou spolupráci a několik klíčových funkcionalit, zbytek funkčnosti dodefinujete a pomocí priorit řídíte až v průběhu projektu.

Zákazník v takových modelech spolupráce obvykle podepisuje jen rámcový kontrakt a zbytek se řeší až v průběhu projektu. Nefixuje se zadání, někdy ani datum. Zákazník má také často možnost se před každým Sprintem rozhodnout, zda bude chtít investovat do dalšího Sprintu nebo už mu produkt stačí v takové podobě, v jaké je. Obsah Product Backlogu tedy není fixní a slouží jako transparentní pohled na priority produktu.

Souvislosti:

(co budete ještě potřebovat zavést)

Sprint Review: Je ideální pro zapojení zákazníka do týmu. Prezentujte funkcionalitu reálnému zákazníkovi, dostanete tak kvalitnější zpětnou vazbu.

Backlog Refinement: zapojuje zákazníka z druhé strany a dává mu možnost ovlivnit, co bude náplní dalšího Sprintu.

Product Owner: je role, která zná zákazníka nejlépe a hájí jeho zájmy. Má na starosti Product Backlog a spolu se zákazníky určuje prioritu. Product Owner je obvykle někdo zevnitř firmy, kdo je zodpovědný za úspěch projektu. Zákazník může být zvenku a je jich obvykle více s různými, vzájemně se vylučujícími zájmy.

Role manažera

Ač si občas některé týmy myslí opak, role manažera má i v agilním světě své místo a překvapivě je možná ještě důležitější než doposud. Moc se o ní ale nemluví. Týmy jsou přece self-organized, tak žádného manažera nepotřebují. Něco pravdy na tom je. Manažera, který se stará o denní operativu, alokuje zdroje na jednotlivé úkoly a kontroluje práci, opravdu nepotřebujeme. To zastane tým sám, a pakliže potřebuje nějak pomoct s každodenními problémy, má Scrum Mastera, který jim pomáhá tyto problémy vyřešit. Scrum Master se také stará o motivaci jednotlivých členů, jejich rozvoj a celkové fungování týmu.

Zodpovědností manažera v agilním světě je primárně vytvoření prostředí, ve kterém agilní týmy můžou dobře fungovat a zlepšovat se, a podpora jednotlivých rolí. V agilních firmách je proto vidět odklon od klasického managera směrem k agilnimu leaderovi, který se stará o rozvoj leadershipu v organizaci a pomáhá Scrum Masterům řešit problémy, na které už jako koučové z nějakého důvodu nestačí. Manažeři můžou zůstat garanty za oblast, kterou měli na starosti (např. vývoj, testování apod.), ale nestarají se již na každodenní bázi o jednotlivé technické úlohy ani jednotlivce. Role manažera se tak posouvá výše, směrem ke strategickým rozhodnutím a denní operativu deleguje na jednotlivé týmy.

Role manažera se v agilním světe mění směrem k agilnímu leadershipu, a je tak ještě důležitější než v klasickém světě.

Agilní organizace od manažerů potřebují jiný styl ledershipu, který podporuje rozvoj leadershipu v rámci organizace.

Jak již bylo zmíněno, není vhodné, když Scrum Master je zároveň i manažerem. Sklouzává pak často k použití síly a direktivního přístupu na úkor role, kterou by jako

Scrum Master měl zastávat. Z klasických teamleaderů se obvykle stávají zpět zkušení členové týmu, kteří mají navíc na starosti danou technologickou oblast jako její garanti, a jsou tak zodpovědni za růst členů týmů v dané oblasti. Přestávají být manažery a nejsou již zodpovědni za jednotlivé lidi, ale jako garanti určité oblasti pomáhají ostatním získat nové zkušenosti a znalosti v dané oblasti. V některých případech se z teamleaderů naopak stávají Scrum Masteři a technologii nechávají na ostatních.

Ať již máte jakoukoli roli, je vhodné se zamyslet, nakolik jste uvnitř týmu, a máte tedy právo zasahovat do organizace každodenních činností týmu, anebo máte spíše roli pozorovatele a necháte tým se organizovat sám.

Role projektového manažera

Role projektového manažera se v agilních firmách stává nepotřebnou, zrovna tak jako samotné projekty. Obojí patří do starého klasického světa. V agilním světě se práce, kterou je potřeba udělat, v klasickém světě definovaná projektem a řízená projektovým manažerem, definuje jako položky backlogu, které v rámci Product Backlogu prioritizuje Product Owner. Ve Scrumu zodpovědnosti projektového manažera přechází v podstatě rovným dílem na Product Ownera, Scrum Mastera a Development tým.

Neznamená to však že by projektoví manažeři přišli o práci. Stávají se Product Owners, Scrum Masters nebo stakeholdery, kteří Scrum týmům pomáhají pochopit potřeby zákazníků a identifikovat business hodnotu. Je to jako s každou pozicí. Role i styl práce se v agilním prostředí mění u všech rolí, znalosti a zkušenosti jsou však stále potřeba.