Zuzana Šochová, Eduard Kunce

AGILNÍ METODY ŘÍZENÍ PROJEKTŮ

computer press

Zuzana Šochová, Eduard Kunce

Agilní metody řízení projektů

Computer Press
Brno
2014

Agilní metody řízení projektů

Zuzana Šochová, Eduard Kunce

Obálka: Martin Sodomka
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN 978-80-251-4194-6

Vydalo nakladatelství Computer Press v Brně roku 2014 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 18 468.

1. vydání

ALBATROS MEDIA a.s.

Obsah

ČÁST IV

PŘÍKLADY

Úvod

Dostává se vám do ruky publikace o agilních metodách. Snažili jsme se čtivou formou popsat všechna možná zákoutí a nástrahy, které vás při přechodu na agilní metody mohou potkat. Kniha nepopisuje pouze to, co to agilní metody jsou, jak funguje Scrum a Kanban a jak by jednotlivé artefakty těchto procesů měly správně vypadat, ale primárně se snaží vysvětlit filozofii agilního přístupu a zaměřit se na vysvětlení, proč jednotlivé metody fungují. Součástí textu jsou praktické příklady a doporučení, co dělat a čeho se naopak vyvarovat.

Kniha se zabývá agilními metodikami vývoje softwaru v různých typech společností, od malých firem až po nadnárodní korporace. V první části knihy jsou popsány základní stavební prvky agilního vývoje, od vlastní definice procesu až po ukázky z praxe. Čtenář se dozví všechny potřebné informace k adopci agilních metodik, kulturní odlišnosti agilních firem a další základní stavební kameny nutné pro úspěšnou adopci agilních principů. Kniha dále obsahuje podrobný popis rolí, které se typicky vyskytují v agilně řízených firmách. Vedle těchto popisů obsahuje kniha i „kuchařky", resp. doporučené postupy adopce agilního řízení pro různé velikosti a typy firem.

V posledních 10 letech se agilní metody staly ve světě velmi populární. Používají je velké nadnárodní korporace jako Amazon, Salesforce, Oracle či Google, najdete je na univerzitách, v telekomunikacích, v bankách a pojišťovnách, v armádě, aerolinkách, automobilovém průmyslu, medicínských firmách a samozřejmě i v mnoha malých či středních firmách. V České republice se dostávají do povědomí firem až v posledních několika letech, ale být agilní a používat Scrum či Kanban již dávno není výjimkou, a agilní metody se tak stávají na českém trhu běžnou metodikou.

Kniha je primárně určena pro lidi pracující v ICT, a to jak pro softwarové vývojáře, testery, designéry, architekty a analytiky, tak i projektové a produktové manažery, vedoucí oddělení, manažery i ředitele či vlastníky takových organizací. Přestože agilní metody byly primárně definované pro zlepšení a zefektivnění vývoje softwaru, používají se v podstatě kdekoli, kde je třeba řídit projekty, organizovat týmy lidí a kde je výsledný produkt komplexní a pro jeho úspěch je vyžadována kreativita, flexibilita, schopnost reagovat na změny (např. v marketingových či reklamních agenturách, konzultantských společnostech nebo hardwarových firmách). Máte-li na starosti nějaký projekt nebo tým lidí, určitě je tato kniha pro vás.

A kladete si hned v začátku otázku, co můžete spolu se zavedením agilních metod očekávat? Určitě vyšší flexibilitu a schopnost reagovat na změny, vyšší efektivitu, kvalitnější produkt, větší předvídatelnost termínu dodání a vyšší spokojenost zákazníka s vaším produktem či službou a v neposlední řadě motivovaný tým, který se spolupodílí na produktu a jeho inovacích. Tedy jednoduše řečeno: být úspěšnější.

Autoři vycházejí z vlastní zkušenosti s nasazováním agilních metodik v různých prostředích firem, od malých startupů až po mezinárodní korporace. Ze svých bohatých zkušeností autoři vybrali vše podstatné a srozumitelnou formou předkládají čtenáři návod, jak postupovat, od prvních krůčků až po kompletní agilní transformaci celé společnosti.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu připravilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih najdete chybu, budeme rádi, pokud nám ji oznámíte. Ostatní uživatele tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2140* po klepnutí na odkaz Soubory ke stažení.

ČÁST I

FILOZOFIE AGILNÍCH A LEAN METOD

- **KAPITOLA 1 –** Agilní a Lean metody
- **KAPITOLA 2 –** Transformace na Agilní metody

KAPITOLA 1

Agilní a Lean metody

V této kapitole:

- Co si představíte pod slovem „agilní“?
- Manifest agilního vývoje softwaru
- Co je to „lean“?

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 filosofií. Přináší to odlišnou firemní kulturu a náladu. 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. Děláme zásadně to, co má v danou chvíli smysl, a děláme to tak, jak nejlépe umíme. Nejde sice o 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

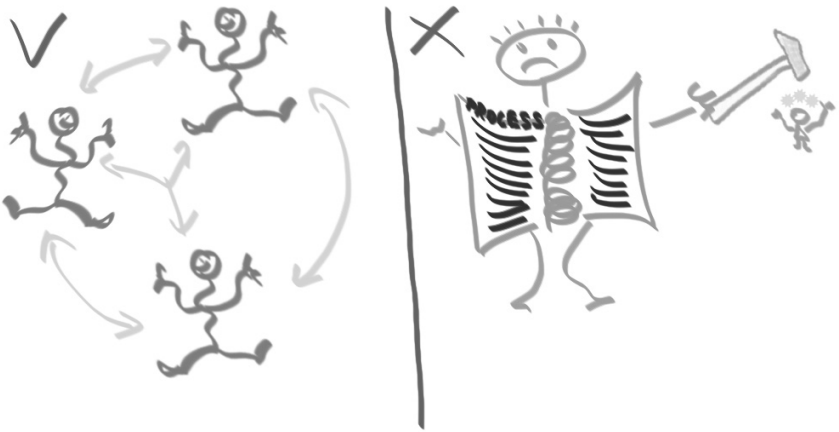
Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více. Pojďme se společně chvíli zamyslet nad tím, co nám vlastně manifest říká.

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. Hlupák i s nejlepším nástrojem je zkrátka pořád hlupák.

Na druhou stranu, manifest neříká nic o tom, že procesy a dohody my 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.

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, z nich každý má aspoň 1 000 stran? 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. Je důležité se už při jejím podpisu zamyslet nad tím, co se při spolupráci může stát. Je třeba si uvědomit, že zákazník možná ne vždy ví přesně, co přesně chce, a že se stejně bude měnit rozsah a smlouvu se 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 požadavky na změny 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 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" je i „lean" spíše o přístupu než striktním procesu. Navíc jsou si oba principy 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 musíme jednotlivé věci předělat, opravit, za-

hodit – 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, které mají smysl, budete jistě efektivnější.
- **Zlepšujte se a učte se již v průběhu práce** – 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.

KAPITOLA 2

Transformace na agilní metody

V této kapitole:

- Vývoj softwaru
- Proč vůbec něco měnit
- Jaké jsou nejčastější důvody pro přechod na agilní metody?

Vývoj softwaru

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 rozpočet (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.

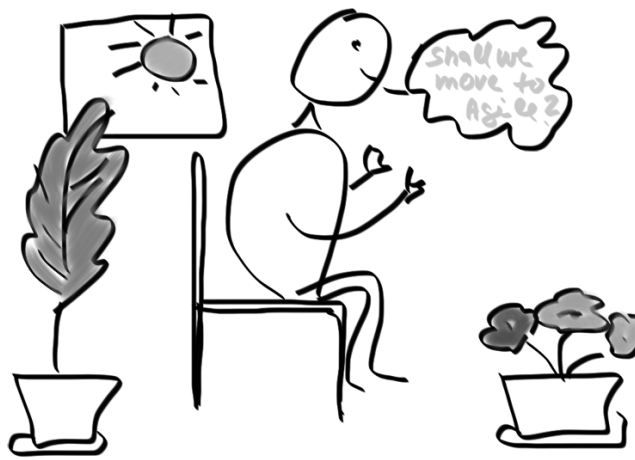
Prvním problémem je, že zákazník často neví, co chce. Tedy vždy si myslí, že ví naprosto přesně, co chce, ale nedokáže to efektivně předat svému dodavateli. Aplikaci si neumí představit, má jen hrubou představu o tom, jak by mu měla pomoci a urychlit či jinak zlepšit práci. A typický zákazník už vůbec 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 neznamená jen změnu 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 – donedávna to šlo. Dělali jste releasy jednou za 6–12 měsíců, zákazníci na to byli zvyklí a firma také. Když chtěli nějakou změnu, protáhli jste jí vaší softwarovou 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 softwarové 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, 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, 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.

Předvídatelnost – jestli vaše projekty končí včas a bez přesčasů před koncem releasu, asi vám váš proces funguje dobře. Pokud to tak není, agilní metody vám mohou pomoci i s tímto aspektem. Tyto metody zavádějí zcela odlišný způsob odhadování. Odhadují v relativních jednotkách a do odhadování zapojují celý tým. Zprvu to zní hodně zvláštně, ale přesnost vašich odhadů se časem výrazně zlepší. Rozdělit projekt na malé kousky je druhou důležitou metodou, jak zlepšit předvídatelnost. Na krátkém úseku se méně projevuje tzv. studentský syndrom a úsilí (effort), které tým investuje, je stabilnější, bez výkyvů a stresu na konci.

Kvalita – tady cílíme na dvě oblasti. Za prvé zapojíme do produktu zákazníka. Zeptáme se ho, co chce a proč a také na to, kdo a jak bude produkt používat. Ukazujeme mu výsledky po malých kouscích a tím řídíme jeho očekávání. Nestává se pak, že by

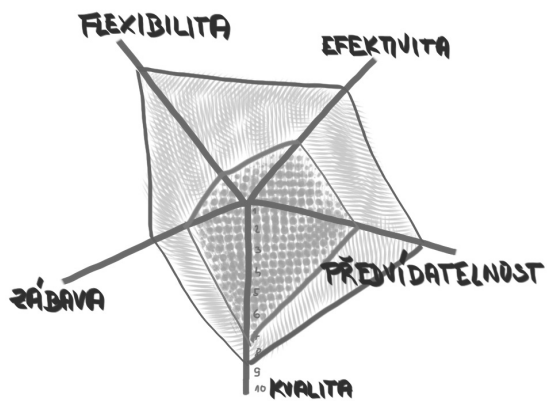
zákazník produkt odmítl jako nepoužitelný a trval na jeho přepsání. Na druhé straně tím, že uděláte celý tým zodpovědný za kvalitu výsledku, se sníží počet chyb a vzroste dlouhodobá udržitelnost kódu. Tým se sám stará o to, že jim neroste technický dluh. Pro oba směry zavádějí agilní metody spoustu praktik.

Zábava – a v neposlední řadě práce bude zase zábava. I jednotliví členové budou vědět, co a proč píšou. Budou chápat smysl produktu a rozumět zákazníkovi. Navíc spolupráce s ostatními nakonec také přináší zábavu. A motivovaný člen týmu je jistě efektivnější než znuděný vývojář sedící sám za svým počítačem.

Poznámka: 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 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.



ČÁST II

POPIS METOD, PROCESŮ, PRAKTIK A ARTEFAKTŮ

- **KAPITOLA 3 –** Agilní slovníček
- **KAPITOLA 4 –** Role
- **KAPITOLA 5 –** Artefakty
- **KAPITOLA 6 –** Meetingy
- **KAPITOLA 7 –** Agilní programování – Artefakty
- **KAPITOLA 8 –** Scrum 1-1 a škálovatelnost Scrumu
- **KAPITOLA 9 –** Extreme Programming 1-1
- **KAPITOLA 10 –** Kanban 1-1
- **KAPITOLA 11 –** Jak funguje tým
- **KAPITOLA 12 –** Smlouvy a formy spolupráce
- **KAPITOLA 13 –** Otázky na závěr

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 následujících sekcích. Teorie je vždy doplněná doporučeními, jak ji v praxi aplikovat. Každá kapitola končí sekcí souvislosti, ve které najdete praktiky, které s popsanou kapitolou úzce souvisejí a které budete potřebovat zavést, aby daná praktika fungovala a přinesla očekávaný efekt.

KAPITOLA 3

Agilní slovníček

Praktiky procesy a artefakty	Vysvětlení
Akceptační kritéria	Doplňuje základní popis User Story o další kritéria definující, jak se pozná, že je User Story hotová.
Backlog Grooming	Pravidelný meeting, na kterém tým probírá s Product Ownerem prioritní User Story z Backlogu tak, aby jim porozuměl a byl je schopen ohodnotit.
Burndown graf	Graf zobrazující rychlost „spalování" funkcionality v Product Backlogu. V závislosti na rychlosti týmu vizualizuje predikci dokončení.
Done kritéria	Domluvená pravidla pro předání, definující hotovou User Story.
Epic	Větší funkční celek, který se následně rozpadá na User Stories.
Extreme Prgramming – 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.
Planning	Ve Scrumu 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.
Pre-planning	Příprava pro planning, kde Product Owner identifikuje priority ze kterých následně tým při planningu vybírá.
Product Backlog	Prioritizovaný seznam funkcionality (user stories), kterou chceme dodat zákazníkovi. Backlog je vytvářený a udržovaný Product Ownerem. Vždy přístupný celému týmu před začátkem, ale i během práce na produktu.

Praktiky procesy a artefakty	Vysvětlení
Product Owner	Vlastník vize produktu. Má na starosti 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.
Product Owner Proxy	Pomocná role Product Ownera u větších produktových týmů, typicky v různých geo-lokacích a časových zónách
Retrospektiva	Týmový meeting, kde jednotliví členové zhodnotí, co se jim v 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 dokončené User Stories.
Rychlost (Velocity)	Rychlost měříme vždy 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.
Scrum	Proces postavený na týmové spolupráci, zapojení zákazníka a pravidelné zpětné vazbě v krátkých Sprintech. Scrum je v současnosti jedna z nejčastěji využívaných agilních metodik.
Scrum Master	Kouč a moderátor týmu. Odstraňuje překážky, stará se o rozvoj a fungování týmu. Udržuje Scrum v chodu. Není manažerem týmu.
Scrum Tabule	Zobrazuje stav aktuálního Sprintu. Obvykle pomocí papírových lístečků s jednotlivými User Stories a tasky.
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.
Standup / Scrum Meeting	Krátký týmový meeting, pravidelně každý den, obvykle ráno. Jednotliví členové sdílí mezi sebou aktuální stav a identifikují případné problémy. Je moderován 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.

Praktiky procesy a artefakty	Vysvětlení
User Story	Funkcionalita popsaná v tzv. kanonické formě: Jako uživatel, chci funkcionalitu, abych dostal hodnotu. User Story má přesně definovaná akceptační kritéria a je dostatečně malá, aby ji tým rozuměl a mohl odhadnout její velikost.

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.
Waterfall	Tradiční metodika vývoje softwaru a jiných projektů spočívající v řetězení činností za sebe, tzn. dokud se nedokončí předchozí fáze, další nemůže začít.

KAPITOLA 4

Role

V této kapitole:

- Scrum Master
- Product Owner
- Self-organized tým
- Scrum tým
- Zákazník
- Product Owner Proxy
- Role Manažera
- Role Project Manažera

Scrum Master

Takže jak vlastně Scrum funguje? 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 není teamleader v klasickém slova smyslu, ale pracuje jako mezičlánek mezi týmem a jakýmkoliv rušivým elementem zvenku. 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 dosáhnout jeho cílů
- odstraňuje problémy
- motivuje tým k lepším výsledkům
- chrání tým před vnějšími vlivy, které by ho mohly odvádět od soustředěné práce na definovaném cíli

Dále se stará se o to, aby Scrum 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 neměl být direktivní.

Scrum Master je ten, kdo „zametá cestičku", aby se týmu dobře 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 kolik Scrum Master jako nákladová položka firmu stojí.

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

Poznámka: Technicky nejzkušenější člen týmu obvykle není ideálním Scrum Masterem.

Scrum Master musí být vnímavý, klást otázky a podporovat schopnost týmu si na řešení problémů přijít sám i za cenu lokální neefektivity.

Expert je naopak člověk, který dokáže poradit a zná obvykle odpověď na otázky lépe než jiní členové.

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.

Role Scrum Mastera je pro úspěch produktu klíčová, vyžaduje vnímavého člověka se schopností dobře komunikovat, koučovat, moderovat. 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 – a o to méně si na konci vzpomenete, že by to mohlo mít souvislost s neexistujícím Scrum Masterem.

Říkali jsme, že Scrum Master nemá být direktivní, není to manažer týmu. Na druhou stranu, nemůže být ale 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 na své problémy 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.

Upozornění: Kombinace rolí vždy přináší rizika a je velmi náročná. Když už jste si jisti, že je nutné role kombinovat, dejte si pozor, abyste rozuměli dobře konfliktům 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: I když se Scrum Master stará o správně fungující tým, bez Product Ownera, který udává týmu směr a priority, se nikdy nedosáhne správného výsledku.

Role Manažera: V agilním světě se vzdaluje každodenní operativě, alokování zdrojů, a organizaci času jednotlivých lidí. To má na starosti tým. Manažer je odpovědný za vytvoření prostředí, ve kterém tým může efektivně pracovat, a podílí se více na strategických rozhodnutích než na řešení každodenních problémů, které si může tým vyřešit spolu se Scrum Masterem sám.

Product Owner

Další z rolí, které Scrum zavádí, je role Product Ownera. Product Owner je vlastníkem produktu. Má na starosti definování vize projektu 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 Value a také ROI celého produktu.

Product Owner je tedy zodpovědný za celý Product Backlog. Není na takovou práci sám, ale obvykle mu pomáhá tým lidí. Kdo by v takovém product týmu mohl být? To záleží hodně na kontextu vašeho produktu. Obvykle tam je někdo z následující skupiny: zástupce zákazníka, uživatelů, softwarový architekt, User Experience, jiní Product Owneři.

Product Owner je týmu pravidelně podle potřeby k dispozici, ale na rozdíl od Scrum Mastera už s týmem ne vždy sedí 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 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 porozumění produktu. Tento cíl následně komunikuje tak, aby všichni věděli, kam jdeme, proč a jak. A to jak v rámci týmu, tak managementu a zákazníků. Cíl musí být pro všechny stejný, i když jazyk, kterým ho do jednotlivých skupin komunikuje, bude pravděpodobně odlišný.

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.

Poznámka: 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 % času 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, a 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.