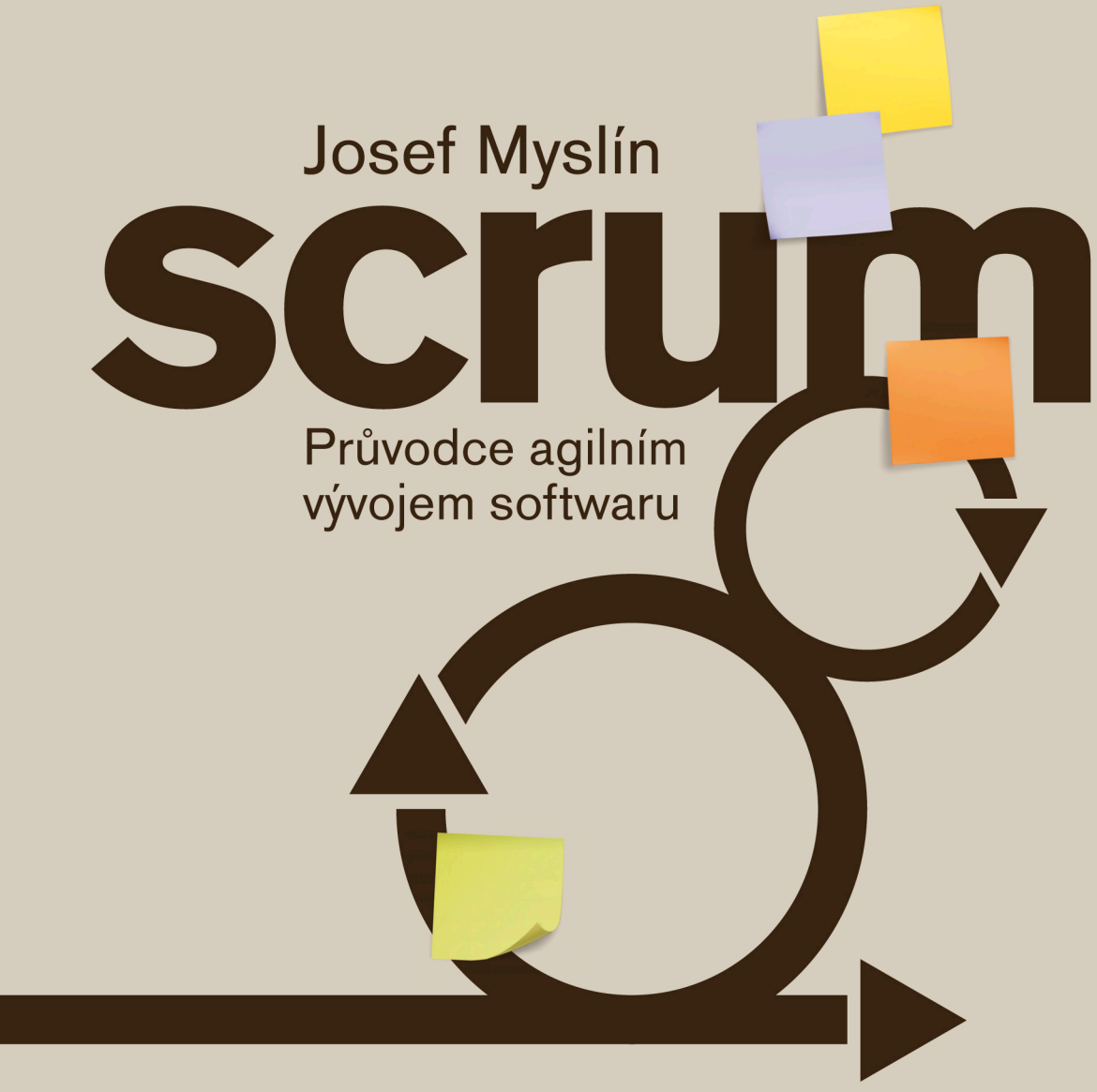
Josef Myslín

scrum

Průvodce agilním vývojem softwaru



computer press

Josef Myslín

Scrum
Průvodce agilním vývojem softwaru

Computer Press
Brno
2016

Scrum
Průvodce agilním vývojem softwaru

Josef Myslín

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 978-80-251-4650-7

Vydalo nakladatelství Computer Press v Brně roku 2016 ve společnosti Albatros Media a. s.
se sídlem Na Pankráci 30, Praha 4. Číslo publikace 23 614.

1. vydání

ALBATROS MEDIA a.s.

Obsah

O čem je tato kniha?

Počítače a další prostředky informačních a komunikačních technologií se již dávno staly nedílnou součástí lidského života. Tak nedílnou, že vyhnout se jim v podstatě znamená rezignovat na výdobytky civilizace. Jsou všude – pokud bychom měli vyjmenovat, pak musíme začít s klasickými počítači v různých kabátech – stolní počítače, notebooky, netbooky, tablety, mobilní telefony. Ale to ani zdaleka není vše. Počítače jsou i tam, kde by je ti méně znalí vůbec nehledali. Jsou v automobilech, jsou v různých domácích spotřebičích, jsou součástí (či někdy spíše podstatou) mnoha lékařských přístrojů. Dnešní dobu můžeme zkrátka označit jako *dobu počítačovou* a nás, moderní lidské bytosti, jako *člověka počítačového*.

O počítačích a dalších zařízeních, které souborně označujeme jako prostředky informačních a komunikačních technologií, však můžeme říci totéž co o ohni. Jsou dobrým sluhou, ale špatným pánem. Jestliže jsou nasazeny správně, uvážlivě a s ohledem na jejich skutečnou roli, dovedou neuvěřitelným způsobem usnadnit život. Jen si zkuste představit psaní této knihy. Dříve bych musel psát knihu (byť na zcela jiné téma, protože počítače neexistovaly) rukou či v lepším případě psacím strojem. Uvažte – žádná možnost uložení a opětovného návratu, žádná možnost snadných oprav, žádná možnost nastavení písma, žádná možnost tisku libovolného množství kopií, žádná možnost zaslání dokumentu někomu jinému prostřednictvím sítě Internet. Sebemenší chyba znamenala nutnost psaní celé stránky od začátku, kopírování bylo, pokud ne přímo nemožné, pak rozhodně velmi obtížné a nepohodlné. A takto bychom mohli s výčtem problémů pokračovat.

Počítač přinesl obrovské změny. I ten nejjednodušší textový procesor, který je obsažen v operačním systému jako jeho integrální součást, nabízí při psaní textu daleko vyšší komfort. Počínaje právě možností ukládat a k uloženému se opět vracet, přes možnost snadných úprav, možnost nastavení písma, stránky a dalších vlastností dokumentu, až po možnost chrlení nekonečného množství kopií. Tady počítač slouží. A slouží dobře. A to nehovořím o pokročilých počítačových aplikacích, jakými jsou například podnikové informační systémy schopné řídit celé velké korporace od A do Z, případně moderní výpočetní systémy schopné řešit složité problémy naší doby.

Ale počítače, respektive jejich špatné či nevhodné nasazení, mohou i škodit. V posledních letech jsme měli možnost zaregistrovat několik systémů, které nejednomu člověku přivodily doslova peklo na zemi. Ať už to byl nefunkční registr motorových vozidel, díky kterému mnozí noví majitelé automobilů či motocyklů doslova kempovali před příslušnými úřady a doufali, že systém během několika hodin opět alespoň na krátkou dobu poběží a umožní jim splnit si jejich zákonnou povinnost přihlásit své auto. Nebo

můžeme zmínit rovněž nefunkční systém mající na starost výplatu sociálních dávek. I zde se nemálo lidí dostalo do velkých potíží, když například nedostali rodičovský příspěvek a nemohli tak zaplatit nezbytné a neodkladné výdaje. Oba systémy nemají náhradu. Pokud nefungují, neexistuje žádný záložní mechanismus, jak lidem umožnit přístup k příslušné službě.

Ale nemusíme jít až tak daleko. Mnohdy počítačová aplikace sama o sobě funguje, a přesto je zdrojem obtíží. To se stává tehdy, je-li aplikace špatně navržena, bez ohledu na reálné charakteristiky zákazníka a konkrétních uživatelů, na jejich potřeby, na jejich schopnosti, na jejich dovednosti, na jejich preference. My „ajťáci" máme poměrně nehezkou vlastnost, za kterou jsme mnohdy okolím odsuzováni – neumíme se vcítit do role běžného uživatele, pro kterého počítač není koníčkem, ale pouhou životní nutností.

Lidé si nekupují počítače proto, že je mají rádi, ale proto, že je potřebují. Nemají touhu se, jak my říkáme, „v počítači hrabat". Touží po jediném – aby mohli snadno a efektivně plnit úkoly, které jsou na ně v pracovním či osobním životě kladeny. Potřebují zaúčtovat fakturu, napsat dopis, přijmout nového pacienta, vyskladnit objednávku, zjistit odjezd toho správného autobusu a mnoho dalšího. A chtějí, aby počítačový systém umožnil vykonávat tyto úkoly snadno a rychle. Jenže reálná situace je často zcela opačná. Počítačové systémy jsou často velmi komplikované a i jednoduché úkoly vyžadují procházení mnoha obrazovkami, mnoho klepání myší. Často jsou postupy zcela nelogické. A důvod je poměrně prostý – vývojář nevytvářel program k obrazu zákazníkovu, ale k obrazu svému. A tak místo toho, aby uživatel v maximální možné míře pracoval tak, jak je zvyklý, a počítač mu pouze asistoval a usnadňoval jeho práci, musí se tento uživatel naopak přizpůsobovat počítači. A to je špatně. Jestliže počítač lidem nepomáhá, pak nemá morální nárok na existenci.

Jenže jak zajistit, aby počítače sloužily a pomáhaly? Každý počítačový systém se skládá z hardwaru a softwaru. Hardwarem rozumíme technické vybavení, to, nač si obvykle můžeme sáhnout. Jsou to všechny ty notebooky, tablety, síťové prvky, monitory, tiskárny, myši, kabeláž. O hardwaru tato kniha určitě nebude. Bude o softwaru, bude o tom, jak vytvářet software tak, aby opravdu odpovídal požadavkům zákazníka. Bude o jedné konkrétní metodice, kterou můžeme použít k tomu, abychom měli nad svým softwarovým projektem kontrolu a abychom byli schopni dovést své projekty ke zdárnému konci, to jest k předání a řádnému používání našeho softwaru zákazníkem. Touto metodikou je metodika SCRUM, které bude věnována převážná část této knihy.

Ale nemůžeme chápat tuto (či některou jinou) metodiku jako něco, co žije v prázdnotě prostoru. Metodika a software samotný jsou součástí širšího kontextu. A právě proto se v této knize zmíním také o obecném pojetí metodik řízení softwaru, o procesu vývoje softwaru, o roli lidí v projektu či například o způsobu měření různých charakteristik projektu. Cílem této knihy pak je, abyste po jejím přečtení byli schopni vnímat softwa-

rový projekt komplexně a abyste byli schopni společně se zákazníkem vytvářet software, který bude opravdu dobře sloužit svému účelu.

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/K2210* po klepnutí na odkaz Soubory ke stažení.

KAPITOLA 1

Proces vývoje softwaru

V této kapitole:

- Co je proces vývoje softwaru?
- Metodiky vývoje software
- Agilní vývoj softwaru

Jestli tato kniha něčím nemá být, pak nemá být teoretickým pojednáním. Problémem ovšem je fakt, že zcela bez teorie se prostě a jednoduše neobejdeme. Nemusíte se však bát, nemám v úmyslu zahltit vás formálními teoretickými definicemi, a pokud se tu a tam něco, co definicí zavání, přece jen objeví, pak udělám vše pro to, abych takovou definici názorně ozřejmil a abych ukázal, o co nám vlastně ve vývojářské praxi jde.

Poznámka: Zde se krátce zamyslím nad tím, proč je teorie tak neoblíbená. Pochopitelně, koho by bavilo biflovat se zpaměti nudné definice, které nijak nesouvisí s praxí. Jenže problémem je, že ty nudné definice povětšinou s tou praxí souvisí opravdu hodně. Jestliže pojedete například autem po mostě, zkuste se zamyslet nad tím, zda by konstruktér toho mostu mohl most naprojektovat, kdyby neznal základní poučky o statice, dynamice, kdyby neovládal základní (z jeho pohledu) matematické úkony a další potřebné věci. Jistě, když člověk začíná, teorie je pro něj neprobádaná oblast, které nerozumí a která se mu nezjevuje v souvislostech. Ty přichází až s mnoha nabytými znalostmi a zkušenostmi.

IT se mění, ať chceme či nikoliv. Zatímco dříve mohl být „ajťákem" člověk, který uměl například dobře programovat či zapojovat sítě, dnes to nestačí. Dnešní doba vyžaduje širší rozhled. IT pracovníci již nejsou ti neznámí, kteří někde ve sklepních kójích sedí a starají se jen o provoz sítě, i když i takoví stále existují a mají své místo. Například se ukazuje jako nezbytné mít určité povědomí o ekonomické stránce věci, o právní stránce věci atd. Rovněž existují různé moderní manažerské metody, které například využívají alespoň základy pokročilejší matematiky. Jestliže jako IT pracovníci budeme chtít zůstat v obraze, budeme se muset s tímto vyrovnat.

Co je proces vývoje softwaru?

Kapitola má název „Proces vývoje softwaru". Proto považuji za nezbytné, abychom si tento pojem vysvětlili. A pojďme na to pěkně postupně. Začněme posledním slovem, slovem **software**. Co je myšleno pojmem software? Tohle slovo zná asi každý, kdo se trošku více zabývá počítači, byť jako lehce poučený uživatel. Software můžeme jinak

nazvat programové vybavení počítače. Tedy jsou to všechny programy, které mohou být v počítači nainstalovány. Pro tuto chvíli můžeme opominout rozdělení na systémové a aplikační programy, protože to pro naše pochopení není důležité. Vše, s čím pracujeme na obrazovkách počítačů, notebooků, tabletů, mobilů a dalších zařízení, je software. Ale i zařízení, kde nevidíme žádný displej či obecně komunikační rozhraní, mohou mít software. Softwaru můžeme také chápat jako duši. Duši, která hmotnému tělu vdechuje život. Bez software by hardware, tedy technické vybavení počítače, bylo jen souborem elektronických součástek, který ovšem není schopen být jakkoliv užitečný. Se softwarem je to ovšem jinak. Software zařídí, že ty elektronické součástky začnou provádět to, co určuje vůle uživatele.

Upozornění: Občas slyšíme, že počítač zase neudělal to, co měl. Tady je ovšem problém. Počítač nemá sebemenší inteligenci. Umí udělat jen přesně to, co měl naprogramováno. Nic více a nic méně. Jestliže tedy udělal něco jiného, než co jsme očekávali, pak musí existovat člověk, který počítači tohle uložil udělat. Vinen je tedy člověk, ne počítač. Pochopitelně nebereme v úvahu situaci, kdy počítač má poruchu.

Víme tedy, co je software (a většina to věděla, aniž by musela předchozí odstavec číst, ale přesto jsem stále přesvědčen, že zde předchozí odstavec své místo má), a můžeme plynule přejít ke druhému slovu. Tím je (zachováme pořadí od konce k začátku, které je trošku neobvyklé, ale v tomto případě výstižné) slovo **vývoj**. Ano, software vyvíjíme, nikoliv tvoříme a nikoliv vyrábíme. Tvořivost patří do umění. Tam si ji nejenže můžeme dovolit, tam si ji dovolit musíme, aby to vůbec bylo umění. Jakmile totiž umění postrádá tvořivost, není uměním, je řemeslem (byť třeba velmi dobrým řemeslem). Ale software není o tvořivosti, my totiž nechceme vytvořit něco, co se líbí, něco, co nadchne davy, něco, co probudí emoce, či něco, co bude vyvolávat ty správné asociace dle umělcova záměru. My chceme vytvořit něco, co především funguje a plní požadavky zákazníka. A míra naší improvizace je tím určena.

Na druhou stranu, u softwaru nesmíme sklouznout ani k té řemeslné rutině. Pokud má software sloužit zákazníkovi dobře, či dokonce výborně, musí být zákazníkovi šit na míru. Nesmíme nikdy sklouznout k tomu, že budeme bezmyšlenkovitě aplikovat to, co jsme použili v předchozím projektu (projektech). Pochopitelně, že můžeme využívat zkušenosti i konkrétní části, konstrukce či komponenty. Ale musíme vždy zvážit, zda takové znovupoužití je v souladu se zájmy projektu, a především se zájmy zákazníka. Pokud chceme vyvíjet kvalitní software, pak z našeho slovníku nikdy nesmí vypadnout otázka „proč“, a naopak do tohoto slovníku nesmí vstoupit odpověď „protože se to takhle dělá“. Vývoj tedy stojí někde mezi tvorbou a výrobou. Od výroby se liší tím, že postrádá rutinu v tom, jaké bude konkrétní využití konkrétních nástrojů. Při výrobě je od počátku více či méně jasno, co má vzniknout. Tohle ve vývoji nemáme. Vyvíjíme něco nového, něco, co zde doposud nebylo. Zkuste si představit výrobu automobilu. Z výrobní linky

sjíždí jedno auto za druhým – a tato auta jsou stále stejná, jsou obrazem určité, předem dané šablony. Když začínáme vývoj takového automobilu, pak přesně nevíme, jaké auto bude na konci vývoje. My jej teprve vyvíjíme. Na rozdíl od umělecké tvorby však tento vývoj probíhá systematicky. Víme, že nejprve přijde jedna fáze, pak druhá, pak třetí. Víme, kdy, jak, co a kým budeme testovat, víme, jak budeme získávat jednotlivé lidi do projektu, víme, jak budeme v projektu komunikovat. Tedy vývoj postrádá rutinu ve výsledku, ale obsahuje rutinu v procesu.

A vida, lehce oklikou jsme se dostali k poslednímu slovíčku, které je potřeba definovat. Tím slůvkem je **proces**. Proces je slovo, které v posledních několika letech zní stále častěji mezi manažery všech úrovní. Procesní řízení je totiž v módě, ačkoliv mnozí vůbec nevědí, o čem je řeč, a mnozí další jen opakují naučené fráze. Co je to tedy ten proces (nemáme nyní na mysli soudní kauzu)? Zde si neodpustím definici, ale jak jsem slíbil, definici důkladně rozeberu, abyste měli jistotu, že opravdu pochopíte podstatu problematiky. Tak tedy – proces je po částech uspořádaná posloupnost aktivit vedoucí od daného počátečního stavu k danému cíli. To zní strašně, viďte? Naštěstí lze tuto větu přeložit do jazyka obyčejných smrtelníků.

Tak za prvé – proces není nic exotického. S procesy běžně pracujeme, pouze jim tak v běžné řeči neříkáme. Jestliže má někdo ve zvyku ráno si připravit šálek kávy, pak můžeme hovořit o procesu přípravy kávy. Nebo můžeme hovořit o procesu oblékání, procesu dopravy do zaměstnání atd. Každý takový proces má určitý počáteční stav a určitý cíl. Tak kupříkladu proces vaření kávy. Ten začíná zcela jednoduše tím, že máme chuť na kávu. To je impulsem, který spouští samotný proces. A proces je ukončen tím, že na stole stojí šálek horké kávy. Mezi počátkem a koncem pak musíme vykonat určité kroky – to je ona zmíněná posloupnost aktivit. Zbývá nám ještě vysvětlit, co znamená po částech uspořádaná. Samotná uspořádanost je asi jasná – kroky, tedy aktivity, mají určené pořadí. Nelze tedy například vypít kávu, která nebyla uvařena. Ale proč po částech uspořádaná? Je to proto, že ona posloupnost nemusí být vždy přísně lineární, že mnohdy musíme sice udělat jednotlivé kroky, ale u mnoha z nich není přesně dáno pořadí. Tak například můžeme nejprve nasypat do hrníčku kávu a poté cukr, ale můžeme také zvolit obrácený postup. Na druhou stranu u aktivit naplnění konvice vodou a zapnutí konvice je pořadí jednoznačně dáno.

Vyspělost procesu vývoje

Nyní je pochopitelně na místě ptát se – jak kvalitní je proces vývoje softwaru, který naše firma využívá? Lze vůbec kvalitu nějak měřit? Kdyby nešlo, byla by zde tato kapitola zcela zbytečná. Vzhledem k tomu, že zde tato kapitola je, můžeme si být jisti, že možnost posouzení vyspělosti procesu existuje. Dokonce máme několik možností – některé jsou jednodušší, jiné mají spíše formu auditu. Zájemce o složitější metody mohu pouze

odkázat na příslušnou literaturu. My se zde seznámíme s jednodušším, zato však velmi výstižným modelem hodnocení vyspělosti softwarového procesu.

Model je běžně znám pod zkratkou **CMM**. Pokud si tuto zkratku budeme chtít rozklíčovat, pak dostaneme slovní spojení **Capability Maturity Model**, tedy model vyspělosti dovedností. Model je, jak už bylo naznačeno, poměrně jednoduchý na pochopení – definuje pět úrovní vyspělosti procesu, přičemž každá tato úroveň popisuje, jak proces na této úrovni vypadá a co je nutno doplnit, aby se proces mohl posunout na vyšší úroveň. V následující tabulce pak naleznete přehled těchto pěti úrovní.

Úroveň	Název	Činnost pro postup
1	Initial (iniciální, úvodní)	Opakované postupy
2	Repeatable (opakovatelná)	Komplexní definice procesu
3	Defined (definovaná)	Měření ukazatelů
4	Measured (měřitelná)	Zpětná vazba a zlepšování
5	Optimizing (optimalizovaná)	

Úroveň 1 – Initial

Pojďme se nyní na jednotlivé úrovně podívat podrobněji. Na úrovni 1 (initial, úvodní) se organizace se svým procesem octne tím, že započne svou existenci. Proto hovoříme o iniciální úrovni, protože pro její dosažení není třeba nic vykonat. Na druhou stranu tato úroveň je velmi nízká a proces na této úrovni de facto není schopen generovat kvalitní software, možná s výjimkou těch nejjednodušších aplikací. Proces na této úrovni není de facto definován. Software je vyvíjen intuitivně (a místo vývoje celý proces spíše připomíná onu dříve zmíněnou uměleckou tvorbu), jednotlivé problémy jsou řešeny ad hoc. U větších projektů velmi brzy dojde k velkým rozporům s očekávanými ukazateli (náklady, čas, kvalita), neboť bez definovaného procesu je velmi obtížné či spíše nemožné jakékoliv smysluplné plánování a následné vyhodnocování plánu. Na této úrovni také nefunguje jakékoliv řízení lidských zdrojů a komunikace se zákazníkem je omezena na aktuální problémy a jejich přímé řešení. Komunikace navíc nemá jednotnou podobu, což může způsobovat závažné problémy. Projekt je tak odsouzen k nezdaru okamžitě po svém začátku, tímto způsobem nelze větší projekt ukočírovat ani finančně, ani časově, ale ani nelze zajistit potřebnou kvalitu. Pro organizaci je proto životně důležité urychleně postupovat na žebříčku úrovní.

Představte si, že dostanete za úkol vypočítat určité množství příkladů určitého typu. Na první úrovni pro vás bude každý příklad samostatným úkolem. Mezi jednotlivými příklady neuvidíte žádnou spojitost a každý příklad tak začínáte řešit zcela od začátku. Příklad nebudete řešit systematicky, nýbrž se vždy budete ad hoc pokoušet najít nějaké řešení.

Úroveň 2 – Repeatable

Postup na druhou úroveň (repeatable, opakovatelná) přitom nemusí být záměrný. Jestliže totiž budete na úrovni 1, pak velmi rychle zjistíte určité best practices, tedy postupy, které se osvědčily. Tyto postupy pak přijmete za své a v dalších projektech, případně v dalších fázích téhož projektu je budete používat. Postupně tak část činností nebude řešena ad hoc, ale bude řešena opakovanými postupy, které vychází ze zkušeností dotyčné organizace či projektového týmu. Ačkoliv to vypadá poměrně rozumně (a oproti první úrovni to skutečně pokrok je), není důvod se radovat. Vyskytuje se zde totiž několik podstatných ale, která je nutno brát v potaz. První ale se týká toho, že takto popsané části procesu nejsou ucelené. Není popsán proces jako takový, pouze existují určité zkušenosti s dílčími částmi. To je sice lepší než nic, ale pro praxi je to stále velmi málo. Další problém spočívá v tom, že nelze hovořit o systematicky popsaném procesu, dokonce ani o jeho částech. Platí, že jsme získali zkušenost, že konkrétní úkol lze splnit konkrétním způsobem. Nic více a nic méně. Nikde není zaručeno, že ten konkrétní způsob je opravdu optimální. Ba spíše naopak. Navíc často neexistuje ani písemný popis řešení.

Na této úrovni dojde k tomu, že v množině příkladů rozpoznáte určité podobnosti a určité opakující se podúkoly s rovněž opakujícími se dílčími řešeními. Nebudete tedy každý příklad řešit zcela ad hoc, ale budete se snažit osvědčené postupy aplikovat. Nejste však schopni říci, zda tyto postupy jsou opravdu optimální, zda jsou opravdu obecné atd. Postup také není systematizován. Některé aspekty počítání jsou přitom stále řešeny ad hoc přístupem.

Poznámka: Ve skutečnosti se přechod z první na druhou úroveň povětšinou děje samovolně a bez úmyslného přičinění vývojového týmu či organizace. Jestliže začínáme absolutně bez systému a jestliže vše řešíme ad hoc, pak v určité fázi docházíme vždy ke zjištění, že určité činnosti lze opakovat a že toto opakování povede ke znatelně lepšímu výsledku. A právě při tomto zjištění přecházíme na úroveň 2 – Repeatable. Přechody na další úrovně už ovšem vyžadují systematickou a záměrnou činnost.

Úroveň 3 – Defined

Třetí úroveň (defined, definovaná) už může být považována za počátek něčeho, čemu se říká systematický vývoj softwaru. Na této úrovni je totiž systematicky popsán celý proces vývoje software. Tedy dva obrovské rozdíly oproti předchozí úrovni. Je popsán celý proces vývoje, nikoliv pouze jeho části. Navíc je proces popsán systematicky a metodicky. Už se nejedná pouze o dílčí postupy, které nějak vycházejí, ale o předem stanovený postup, o kterém se přemýšlelo a který je v nějakém smyslu rozumný či optimální. V dnešní době je velmi rozšířená snaha firem získat certifikát o řízení systému jakosti podle normy ISO 9001 – tento certifikát mimo jiné předpokládá právě komplexní popis procesů v dané firmě. Není samozřejmě možné tvrdit, že být na úrovni 3 modelu CMM a získat certifikát podle normy ISO 9001 je totéž. To opravdu není pravda. Nicméně lze

nalézt určité analogie a je evidentní, že podrobný a systematický popis procesu je nutnou (ne však postačující, použijeme-li matematickou terminologii) podmínkou kvality.

Upozornění: Certifikace systému jakosti podle normy ISO 9001 (název normy je ve skutečnosti o něco delší a složitější, ale pro naše potřeby stačí pouhé pochopení) bohužel podlehla určité degradaci a devalvaci. Skutečným cílem této certifikace ve skutečnosti není ona certifikace, ale onen systém řízení jakosti, o kterém norma pojednává. Ve skutečnosti vznikla poptávka po certifikaci (mnoha výběrových řízení se firma bez certifikace ani nemůže zúčastnit). A tak firmy usilují o certifikaci, nikoliv o skutečné reálné zavedení systému řízení jakosti. Ten totiž předpokládá skutečné řízení, ne formální „hození na papír" a divadelní představení při certifikaci. Občas se tento postup povede a výsledkem je, že máme firmu s certifikací, ale bez systému.

Na úrovni 3 máme pro daný typ příkladů přesně stanovený postup výpočtu, který počítá se všemi známými a možnými eventualitami. Postup je ověřen, je korektní, je rozumně popsán tak, aby bylo možné jej rutinně používat. Postup je komplexní a univerzální, tzn. platí pro celou třídu problémů daného typu.

Úroveň 4 – Measured

Třetí úroveň je obecně považována za rozumnou úroveň vyspělosti. Ve skutečnosti je to první úroveň, u které lze konstatovat, že může být dostačující pro vývoj moderního softwaru. Moderní software znamená rozsáhlý informační systém, často propojený s mnoha dalšími systémy softwarovými i hardwarovými, založený na komplikovaných procesech. Moderní software je povětšinou vyvíjen většími týmy a nezřídka týmy z různých společností. A v takovém prostředí nemá vývojář jinou možnost, než postupovat systematicky a metodicky – zkrátka postupovat s využitím inženýrských metod. Třetí úroveň vyspělosti tak není v takovém případě jakousi úrovní středovou, nýbrž úrovní minimální. Je to tak – teprve na třetí úrovni je váš vývojářský tým připraven vyvíjet rozsáhlý moderní software. Nenechte se zmást tím, že se slušný software občas povede i týmům, které postupují vysloveně nemetodicky. Jednak je na zvážení, zda se opravdu jednalo o rozsáhlý sofistikovaný systém, ale především – zda se nejednalo o pouhou náhodu.

Poznámka: Pokud se dítěti podaří nastartovat auto, rozjet se, a dokonce bezpečně zastavit, jistě to nebudeme považovat za doklad způsobilosti řídit motorové vozidlo. A budeme trvat na systematické přípravě v autoškole, při splnění věkových a dalších limitů. Náhoda zkrátka není dobrým společníkem, a pokud existuje jiná cesta, je vhodné se spoléhání na náhodu vyhnout – v tomto případě právě již zmíněným inženýrským přístupem.

Přesto však přístupu, který je popsán na třetí úrovni vyspělosti, cosi chybí. A to něco je měření a vyhodnocení. Je náš postup skutečně efektivní? Nezpůsobuje zbytečné chyby, zbytečné ztráty? Je skutečně jediný možný? A pokud ne, nejsou některé jiné postupy

v něčem lepší? To jsou zcela zásadní otázky, které mohou mít zásadní dopad na úspěch (či bohužel také neúspěch) našich softwarových projektů. Jenže pokud na tyto otázky chceme odpovědět lépe než pouhým hádáním, potřebujeme mít jistá tvrdá data. A právě tato skutečnost je tím, co nás z úrovně třetí posouvá na úroveň čtvrtou. Existence určitých charakteristik projektu, které následně měříme a vyhodnocujeme. Těmito charakteristikami může být v podstatě cokoliv, co je objektivně měřitelné – procento implementovaných scénářů užití, procento úspěšných testů a mnohé další.

Těchto a podobných dalších měřitelných ukazatelů můžeme nalézt v odborné literatuře desítky, ne-li přímo stovky. Nesčetné množství dalších si můžete vymyslet v rámci svého konkrétního projektu. Jediným kritériem by mělo být to, zda daný ukazatel pro vás má nějaký smysl. Neexistuje důvod měřit něco, co neshledáváte sami pro sebe či pro svůj projekt prospěšné.

Poznámka: Zde jen pozor s hodnocením prospěšnosti. S narůstajícími zkušenostmi a znalostmi budete schopni lépe posoudit užitečnost toho či onoho postupu či přístupu – v tomto případě toho či onoho ukazatele. Dbejte na to, abyste k hodnocení užitečnosti nepřistupovali sebestředně či alibisticky – například dle toho, jak jsou jednotlivé ukazatele snadno měřitelné či snadno vyhodnotitelné. Přínosnost by měla být hodnocena zejména ve smyslu přínosu projektu, přínosu týmu, přínosu zákazníkovi. Na to se obecně často zapomíná a přínos se hodnotí individuálně a krátkodobě – na to doplácí například dokumentace, o čemž koneckonců bude pojednáno v jedné z dalších kapitol.

Jestliže tedy jste schopni definovat sadu ukazatelů (část z nich může být zcela obecná, část pak může být závislá na konkrétním projektu a jeho charakteristikách), u těchto ukazatelů pak definovat měřicí stupnice a provádět pravidelná systematická měření a tato následně rozumně vyhodnocovat, pak můžete hovořit o tom, že vyspělost vašeho softwarového procesu se nachází na čtvrté úrovni.

Upozornění: A zde se musíme zastavit. Ve skutečnosti to, že jsme na čtvrté úrovni, neznamená, že měříme charakteristické ukazatele jednoho projektu, abychom zjistili, zda jsme na tom v tomto projektu dobře či špatně. To samozřejmě provádíme také, ale nesvědčí to o vyspělosti softwarového procesu. Nám v tomto případě jde o měření toho, zda samotný proces, který používáme, je správný. Jestliže zjistíme, že se v projektu například odchylujeme od dohodnutého termínu dokončení, je to pro daný projekt sice velmi nepříjemné, ale naprosto nic to neříká o samotném procesu.

Může být chyba v našich pracovnících, kteří nezvládají své úkoly, může být chyba v komunikaci zákazníka, který adekvátně nereaguje na vzniklé problémy, mohly se vyskytnout chyby z vyšší moci. Nás však (z hlediska vyspělosti procesu) zajímá, jak se tyto ukazatele budou chovat ve více projektech či v různých fázích těchto projektů. Tedy jedině s větším množstvím systematicky získaných hodnot ukazatelů můžeme získat rozumné závěry o samotné kvalitě procesu.

Když se vrátíme k našemu případu s řidičem automobilu. Jestliže se testovaná osoba dokáže rozjet s automobilem, pak to samozřejmě nic neříká o tom, zda automobil opravdu umí ovládat. Proto opakovaně testujeme, zda frekventant autoškoly dokáže uvést automobil do stavu s nenulovou rychlostí, proto tuto skutečnost testujeme v různých podmínkách – z kopce, do kopce, na mokré vozovce, na náledí. Teprve tato opakovaná měření nám řeknou, zda samotný proces (v tomto případě postup rozjíždění se) je v pořádku.

Pro naše počítání příkladů bychom na čtvrté úrovni znali nejen postup samotného výpočtu, ale dokázali bychom hodnotit, zda náš postup skutečně vede ke správnému a úplnému výsledku, zda je tento postup efektivní, zda jej nelze nahradit jiným postupem, který by ke kýženému výsledku vedl například s vynaložením menšího úsilí.

Úroveň 5 – Optimizing

Může se zdát, že nyní už máme opravdu vše, co potřebujeme pro správný vývoj aplikací (či jinou činnost – metoda CMM byla sice navržena právě pro SW projekty, ale s menšími či většími modifikacemi je využitelná i v jiných oblastech lidské činnosti). Ale přesto ještě nejsme na konci naší cesty za kvalitním a vyspělým procesem vývoje softwaru. Co nám ještě chybí? Celý proces je podrobně popsán pomocí některé metody popisu či modelování procesů.

V celém procesu máme nastavené ukazatele, které nám dovolí hodnotit, zda daný proces je dobře nastaven, či nikoliv. Kde tedy můžeme ještě najít rezervy? Odpověď je poměrně prostá – k čemu nám je měření, pakliže toto měření nevede ke zlepšování? Dobře, naměřili jsme, že ten a ten ukazatel dosahuje té a té hodnoty. Dokonce jsme dokázali interpretovat tento stav tak, že jsme diagnostikovali určitou neefektivnost našeho procesu. S tím se však musí něco udělat. Jinak měření nemá smysl.

Upozornění: A opět je důležité pochopit, že to, co nás posune na pátou úroveň, není opravení chyby. Ve skutečnosti v podstatě každý, kdo objeví chybu, se pokusí o její nápravu. Nechme nyní stranou to, zda je tento pokus o nápravu krokem správným či špatným směrnem. Ale chybu nenechá jen tak asi nikdo. Jenže pátá úroveň se neliší pokusem opravit chybu. Pátá úroveň spočívá v tom, že v procesu jsou vytvořeny automatické mechanismy, které vyhodnocují situaci a v případě, že se začnou objevovat nesrovnalosti, začnou s nápravou. Náprava chyby není pouhé zoufalé jednání, ale je nedílnou, integrální součástí procesu.

Často se využívají principy negativní či pozitivní zpětné vazby a podobné mechanismy. V případě, že tyto nastavené mechanismy jsou nastaveny (a také používány) správně a korektně, nedochází obvykle ke stavům kritického selhání. Proč? Protože selhávání je obvykle odhaleno již ve chvíli, kdy odchylka reálného stavu od očekávaného je ještě malá. A tato malá odchylka způsobuje žádné či jen velmi malé problémy. A díky korekčním mechanismům je co nejdříve odstraněna.