STRUČNÝ NÁSTIN ZRÁNÍ TVORBY IS

Jiří Peška

Heslo společnosti pro záchranu tonoucích:
Záchrana tonoucích
je věcí tonoucích samých.
Ilja Ilf, Jevgenij Petrov: Dvanáct křesel

ÚVOD

Softwarové produkty jsou vyvíjeny dostatečně dlouho, aby bylo možno hovořit o zrání procesu jejich tvorby. Tento text uvádí něco z toho, co se o zrání tvorby sofwaru objevuje v cizích časopisech a co snad může být pro někoho podnětem.

Na závěr nemastného úvodu uvedu zčistajasna myšlenku, která by se měla táhnout článkem coby výrazná nit. Pokud však o ní nevíte dopředu, nevšimnete si jí.

Podstatné je překlenout mezeru mezi programováním jako uměním a mezi programováním jako řemeslem.

PŘEDPOKLÁDEJME, ŽE CHCEME SOFTWARE JAKOSTNÍ

Efektivní řízení jakosti vyžaduje jasné definice a explicitní měření jakosti. Přitom jsou rozhodující požadavky zákazníků. Zde selhává řada organizací. Nedokážou totiž pojmy svých zákazníků použít ani pro definování jakosti ani pro stanovení měření. Jedná se o typická selhání komunikace. Jedná se o selhávání komunikace, která přicházejí tuze draze, protože informační systémy vytvořené v podmínkách nefungující komunikace zdaleka nepřinášejí, co by měly, spíše daleko odnášejí, co by neměly (důvěru).

Definice jakosti IS musí zahrnovat zákazníkovo pojetí užitečnosti a použitelnosti. *Užitečnost* znamená, nakolik pomáhá aplikace uživateli. *Použitelnost* je kombinací takových atributů, jako jsou uživatelská přívětivost a spolehlivost. Každá definice jakosti musí mít dva rozměry, které jsou často reprezentovány otázkami:

1. Děláme správné věci?
2. Děláme věci správně?

KDO NEMĚŘÍ, NEŘÍDÍ

Dobře definovaná jakost je měřitelná.

Není cílem měření znepříjemňovat lidem život. Právě naopak. Měření zaměřené na zlepšení procesu může zlepšit morálku pracovníků a přinést jim lepší uspokojení z práce. Účinné měření není měření lidí. Měří se práce udělaná lidmi a výsledky slouží ke zlepšení produktu a procesu. Z hlediska efektivnosti je přitom důležité věnovat co nejvíce pozornosti prvním fázím vývoje, protože chyby se vývojem prodražují a jejich náprava na konci procesu je mnohonásobně dražší než na začátku (až 100 krát). Zásadou je porozumět požadavkům zákazníka, aby chyby nebyly již v návrhu.

Každá úroveň organizační hierarchie má své potřeby informací. Proto je třeba, aby podnikové informace byly hierarchicky strukturovány. Vztahuje se to i na měření. Měření a metriky by měly být vytvářeny hierarchicky. Sada měr by měla být konstruována s ohledem na kulturu a cíle organizace tak, aby sloužila jako kritický indikátor činnosti umožňující rychle zjistit, jak organizace pracuje.

Nyní nastal čas, abychom se při diskusi o měření vrátili ke klíčovým otázkám definice jakosti:

- děláme věci správně?
- děláme správné věci?

Těmto dvěma otázkám odpovídají dva druhy měření.

1. Měření procesu a produktu směřuje k tomu, abychom dělali věci správně

Dělání věcí správně znamená, že s nejvyšší mírou efektivity vytváříme jakostní produkty.

Měření produktu a procesu nám mají pomoci odpovědět na otázky: Zlepšují se věci? Proč? Je naše úsilí o zlepšení efektivní?

Odpovědi vyplývající z porovnání probíhajícího projektu s historickými daty mají odhalit faktory vedoucí ke zlepšení.

2. Měření obchodních výsledků směřuje k tomu, abychom dělali správné věci.

Produkty IS a služby vytvářejí obchodní hodnoty. Měření jsou odvozena z obchodní výkonnosti. Změny mohou být měřeny porovnáním počáteční výkonnosti s výkonností dosaženou IS. Přitom je třeba brát v úvahu, že podniky směřují své úsilí do různý oblastí, jako například: návratnost investic, informace pro management, konkurenční výhody, produkční výkonnost. Míry se odvozují podle zvolené oblasti. Příslušná měření pak přinášejí data nastiňující globální obraz stavu: kolik přináší IS zákaznické organizaci, o kolik stoupne hodnota IS zvýšením jakosti.

MODEL ZRÁNÍ

Existuje několik modelů zrání softwarových procesů. Jeden ze známějších modelů byl vytvořen v Software Engineering Institute (SEI) Carnegie Mellon University. Model má pět úrovní (stejně tolik úrovní má i několik dalších modelů).

1. úroveň Počáteční
2. úroveň Opakovatelná
3. úroveň Definovaná
4. úroveň Řízená
5. úroveň Optimalizovaná

Snad už z názvů úrovní lze usoudit, že 1. úroveň je ta nejnižší (zajišťuje pouze dostatečná rizika) a 5. úroveň nejvyšší (poskytuje dostatečné řízení produktivity a jakosti).

V následujícím přehledu jsou úrovně zralosti charakterizovány a jsou uvedeny úkoly, které je vhodné na dané úrovni řešit, abychom se dostali na úroveň vyšší.

Charakteristika	Úkoly
1. Ad hoc/ chaotické procesy	
- Neexistují formální procedury, odhady nákladů, plány projektů	- Řízení projektů - Plánování projektů
- Neexistují mechanizmy řízení zajišťující dodržování postupů	- Řízení konfigurace - Zajišťování jakosti software
2. Intuitivní	
- Proces závisí na jednotlivcích	- Školení
- Vytvořeny základy řízení projektu	- Technické praktiky (přezkoumání,testy)
- Silné jsou procesy, které se opakují, v nových jsou rizika	- Zaměření se na procesy (standardy)
3. Kvalitativní	
- Procesy definované a institucializované	- Měření procesů - Analýza procesů
- Zlepšování procesů je řízeno jakostí	- Kvantitativní plány
4. Kvantitativní	
- Procesy měřené	- Analýza problémů
- Vytvořena základní sada měr jakosti a produktivity	- Předcházení problémům

- Vytvořena databáze procesů	- Změny technologie
5. ...	
- Automatizované shromažďování dat o procesech - Přísná analýza příčin chyb, předcházení chyb	- Udržování úrovně - Procesy stále ještě příliš náročné na lidskou činnost

KOMENTÁŘ K PŘEHLEDU

Úroveň 1.

Cílem je prostě vyvinout software. Postupuje se ad hoc. Nelze určit termíny, náklady, jakost. Pokud se dělají nějaká měření, je to až po dodání produktu zákazníkům. Výsledky dodatečných měření jsou často neradostné.

Typická měření:

- závady zjištěné uživatelem
- velikost aplikace, termín dodávky
- celkové náklady na vývoj
- celkový čas na vývoj.

Úroveň 2.

Cílem na této úrovni je řízení projektu. Často se k tomu užívá kontrola plánem. Řízení nákladů je obtížné, protože se dělá všechno proto, aby produkt byl hotov včas. K dispozici nebývají historická data a není tedy možné vytvořit rozumný plán. Funkčnost a jakost přicházejí nezřídka zkrátka.

Typická měření:

- frekvence problémů
- skutečné termíny dohotovení proti plánovaným
- skutečné náklady proti plánovaným
- počet požadovaných změn
- růst velikosti aplikace.

Úroveň 3.

Cílem je řízení produktu. Produkt splňuje všechny požadavky a jakost je řízena. Náklady však nejsou pod kontrolou a často velmi narůstají zjišťováním chyb a jejich odstraňováním.

Typická měření:

- efektivnost odhalování chyb
- frekvence chyb podle typu
- roztřídění požadavků na změny podle typu.

Úroveň 4.

Cílem je řízení procesu, jímž je produkt vytvářen. Pouze řízením procesu mohou být řízeny náklady a spokojenost zákazníka. Produkt je dodáván v plánovaném čase s plánovanými náklady. Nyní může být proces zlepšován. Mohou být zkracovány časy, snižovány náklady, vytvářeny lepší produkty.

Typická měření:

- efektivnost odstraňování chyb
- měření přepracovaných částí.

Úroveň 5.

Probíhá kontinuální zlepšování. Cílem je optimalizace jakosti i produktivity.

Typická měření:
- aktuální versus předpokládané výsledky zlepšování procesu
- redukce odchylek procesu
- aktuální versus plánované efekty inovace procesu.

Poznámka: To, co je uváděno jako typická měření, slouží jen jako příklad. Nejde o reprezentativní přehled.

Některá měření jsou určena pro kontinuální sledování, jiná slouží jako varovné signály. Při přechodu na vyšší úroveň se stávají dominantními nová měření, stará se stanou varovnými indikátory.

Zavádění měření do organizace představuje změnu kultury. Bývá to často doprovázeno stresem pracovníků. Přirozenou reakcí je, že lidé produkují to, co je měřeno. Měří-li se řádky zdrojového programu, budou produkovat řádky.

Měření nemá být zaměřeno na jednotlivce, ale na celek. Úspěch může nastat pouze, uzná-li potřebu změn organizace jako celek.

CESTA KE ZRALOSTI SOFTWAROVÉHO PROCESU

Pěkným příkladem úspěšného zrání softwarového procesu je vývoj software pro raketoplán [1].

Opomeneme zcela organizační a další nezbytné a zajímavé záležitosti a soustředíme se pouze na půvab postrádající přehled, ilustrující fakt, že zrání vyžaduje čas a úsilí.

V šedesátých letech byly v Houstonu zvládnuty všechny principy tvorby sofware, které byly v té době známy: techniky řízení, softwarové inženýrství, posilování tvořivosti lidí, kultura projevující se péčí o jakost atd. Procesy byly řízeny a jejich nezbytnou součástí byla disciplína, odpovědnost a osobní motivace. Značná pozornost byla věnována požadavkům zákazníků. Pětadvacet let uplatňování a rozvíjení těchto prvků znamenalo úspěch.

Nelze opomenout ani striktní uplatňování konfiguračního řízení, které má pro vývoj velkých, složitých softwarových systémů zásadní důležitost.

Počátek vývoje software pro raketoplán lze klást na začátek sedmdesátých let. Byl vybudován silný management pro vývoj software a útvar pro přezkoumávání architektury software, v němž byli zástupci každé vývojové části projektu. Uživala se měření pro sledování plánů a nákladů. Měření jakosti bylo v plenkách.

Houston však systematicky sbíral a uchovával data o všech problémech. Každý problém musel být vysvětlen NASA. Bylo třeba objasnit, proč byla chyba udělána, existují-li jiné podobné problémy v software a co bylo učiněno, aby se předešlo stejnému druhu chyb v budoucnosti. Systematicky sbíraná data se stala základem pro odhady spolehlivosti a výzkum softwarových metrik.

V osmdesátých letech, kdy jednou z velmi prosazovaných tendencí byla včasná detekce chyb, patřilo k metrikám pro sledování projektů:
- procento včasně zachycených chyb
- chyby v procesu (při testování)
- chyby v produktu.

Bylo důležité sledovat trendy, které se uplatňovaly.

Vzrůst procenta včasně zachycených chyb a pokles chyb v procesu a produktu potvrzoval, že trend je správný. Pokud by například klesal počet chyb v procesu a vzrůstal počet chyb v produktu, bylo by třeba proces prozkoumat, aby se zjistilo, co je příčinou toho, že produkt je dodáván s chybami.

Při tvorbě software pro raketoplán se v sedmdesátých letech včasně zjištěné chyby podílely na všech chybách padesáti procenty. V osmdesátých letech tento podíl vzrostl na 80 %. Příčinou

úspěchu bylo především zavedení inspekcí do jednotlivých fází cyklu vývoje. Inspekcí fáze stanovení požadavků a fáze testování se účastnil zákazník.

Díky neustále shromažďovaným datům může být v Houstonu nyní předvídán počet chyb, které budou nalezeny. Při testování je nyní zjišťováno méně než 10 % chyb. 85 % chyb je odhaleno inspekcemi začleněnými v procesu.

Chyby v produktu jsou prakticky na nule. V roce 1986 bylo 0,72 chyb na 1000 řádek kódu, v roce 1993 to bylo 0,3 chyb na 1000 řádek.

Tím byl ukončen stručný přehled zrání jednoho software. Mnohé potřebné v té zkratce přišlo velmi zkrátka. Ilustrace však měla pouze potvrdit, že zrání vyžaduje čas a úsilí a že základem vlastního systému měření je sběr dat.

Proces prožitý v Houstonu byl přizpůsoben na jiné projekty, takže může být s úspěchem použit i v malých softwarových týmech. Tým si může zvolit úroveň, na niž se chce pohybovat. Postupy jsou využívány např. pro systémy řízení leteckého provozu.

Mnohé poučení lze načerpat v literartuře i z vývoje operačního systému pro AS/400, počítač firmy IBM. První vydání operačního systému v roce 1988 mělo 7,1 milionu řádek zdrojového kódu. Typické nové vydání obsahuje kolem dvou milionů řádek nového a změněného zdrojového kódu. Za nejdůležitější prvek při řízení jakosti jsou považováni lidé. K nejdůležitějším měřením patří spokojenost zákazníka.

ZÁVĚR

V literatuře existuje mnoho různých technik a doporučení. Nic z toho sice nepředstavuje kouzelnou střelu, která by vyřešila vše jednou ranou, ale i tak by základní postupy měl mít český programátor v češtině stále u ruky. Tady je jedna z možností pro zmenšení mezery mezi programováním jako uměním zasvěcených a programovaním jako řemeslem.

LITERATURA

[1] Billings C., Clifton J., Kolkhorst B., Lee E., Wingert W. B.: Journey to mature software process. IBM Systems Journal, Vol. 33, No. 1, 1994.

[2] Goodman P.: Practical Implementation of Software Metrics. McGraw-Hill, 1993.

[3] Keuffel G: A Metrics Reading List. Software Development, April 1995.

[4] Walrad C., Moss E.: Measurement: The key to application development quality. IBM Systems Journal, Vol. 32, No. 3, 1993.

Autor: RNDr. Jiří Peška
OKD Automatizace řízení, a.s.
Gregorova 3
729 41 Ostrava 1
tel: 069 - 626 2789,
fax: 069 - 622 5827