Na co by si měli naši tvůrci softwaru zvyknout

Jiří Šlegr

Motto: *„Do It Right the First Time."*
„Detect It Early, Fix It as Soon as Practicable."
„Monitor It."
ANSI/IEEE Std 982.2–1988

1 Něco úvodem

V loňském příspěvku [1] byl podán nástin toho, jak si normy ANSI/IEEE představují metodiku tvorby softwarových produktů a byly srovnány typické naše postupy s tímto vzorem. Letošní příspěvek chce předložit našim tvůrcům softwaru několik návrhů, které mají původ v tomto srovnání a autor má za to, že jsou nejen rozumným krokem v procesu navazování spolupráce se světem, ale i příspěvkem ke zvýšení kvality naší vlastní práce.

Není snahou tvrdit, že ve světě tvůrci softwaru aplikují tyto normy automaticky a beze zbytku, skutečnosti blíže spíše bude, že si z nich vybírají to, co je pro jejich práci relevantní a co jejich práci pomáhá. Skutečností však zůstává, že jsou velké rozdíly v tom, co si v tomto pojetí myslí naši a co zahraniční tvůrci. Ze styku s těmi zahraničními je patrno, že autoritu norem uznávají, berou je jako nástroj, který tvůrcům dává větší jistotu dosažení cíle, a pomáhá předcházet pozdějším obtížím a nedorozuměním. Jiný postup je pro ně dost těžko pochopitelný. To si autor uvědomil, když zjistil, jak nelehkým úkolem je popsat zahraničním pracovníkům jak při tvorbě softwaru postupujeme my a vysvětlit, proč to děláme právě tak, jak jsme si zvykli.

Zahraničí tvůrci neočekávají, že jim zadavatel předem nevratně věnuje značné prostředky na úkol, který má dát za několik let jakési výsledky. Zde je předem celkem jasné, že po tak dlouhé době už ani zadavateli na výsledcích tolik záležet nebude, protože sotva budou aktuální. Vynaložené prostředky také v tu dobu budou spotřebovány a sotva bude reálné je získat zpět. Takový způsob zadávání prací nemá velkou budoucnost v tržním hospodářství a je reálnější předpokládat, že naopak našim tvůrcům se budou měnit podmínky tak, že stále častěji budou pracovat něco jako na úvěr a řádně zaplaceno dostanou až za výsledky, o kterých se prokáže, že odpovídají požadavkům zadavatele. Zda takové jsou, však nebude rozhodovat sám řešitel, ale zejména zadavatel. Arbitrem zde bude specifikace požadavků na vyvíjený softwarový produkt. Pokud v ní bude něco jako „podle pokynů zadavatele", jak jsme zatím mnohdy zvyklí, ukáže se jako nepoužitelná a v příštím projektu už bude vypadat jinak. O specifikaci požadavků na software je

v tomto příspěvku dále. Názory, že specifikace a podobné dokumenty jsou zavrženíhodnou administrativou po takových zkušenostech dlouho přežívat nebudou. Jaké si tedy vzít ponaučení?

2 Snažte se, aby tvorba softwaru fungovala jako systém

Domácí softwaroví tvůrci, kteří mají rádi konkrétní práci, mívají všeobecný odpor k administrativě; určitě také proto, že ta slavila velké úspěchy, když nám v rozumné práci, často neinteligentním způsobem, bránila. Při tom jsme byli ochotni takové překážky překonávat a dosahovat při tom slušných výsledků. Tím jsme ovšem bezesporu blíže k situaci, kterou lépe charakterizuje výrok (A):„fungují zdatní jedinci", než výrok (B):„funguje systém tvorby programů", který je v názvu této kapitoly.

Následky tohoto stavu jsou zřejmé: Softwarový projekt stojí na zdatných jedincích a přejímá i jejich vlastnosti. Typické je, že chybí dokumentace a je až neuvěřitelné, že u některých projektů skoro jakákoliv a je nahrazena ústním podáním. Mnohé vlastnosti výsledného produktu jsou dány tím, co jsou řešitelé schopni vytvořit a také tím, co stihnou udělat dříve, než je nutno je převést na další úkol. Organizace se snaží zachránit situaci tím, že si „drží" příslušného pracovníka. To je ale pro ni stále obtížnější, pokud je pracovník opravdu schopný.

Lze očekávat stále větší tlak na to, aby pravdivý byl výrok (B), protože je spolehlivější zárukou dobrých výsledků. Proto, v duchu názvu kapitoly, lze tvůrcům softwaru mimo jiné doporučit:

2.1 Tvorba softwaru je proces který je třeba kvalifikovaně řídit

Proto má mít každý projekt na vývoj softwaru svého manažera, který je jmenován na celou dobu, po kterou se projekt realizuje. Není to úředník, který sleduje plnění termínů, ale organizačně schopný odborník, který si vydělává „tvrdou řídící prací" (vyjádření zahraničního odborníka). Na jeho kvalitách z tohoto hlediska velmi záleží jaký bude výsledný produkt, jak jsme vícekráte slyšeli od zahraničních softwarových manažerů. Podle rozsahu a závažnosti projektu má manažer k dispozici další složky, které jsou vytvořeny a personálně obsazeny tak, aby fungovaly, a to adekvátně k rozsahu, charakteru a závažnosti projektu.

2.2 Věnujte systematickou péči kvalitě vyvíjeného softwaru

Pojem „řízení kvality", který známe z minulosti, nepůsobí dobrým dojmem. Jako kdyby měl řídit chod prací tak, aby kvalita byla ne horší, než plánovaná, ale také ne lepší než plánovaná. Mnohem lépe zní anglické „Software quality assurance (SQA)", nebo dejme tomu i české „péče o kvalitu", či něco podobného.

2.2.1 Co je kvalita v případě softwaru?

Asi nepochybujeme, co je míněno kvalitou u hmotných výrobků. Ve světě však nepochybují ani co je tím míněno v případě softwaru a mají pro to nejen termín, ale i normu, která se zabývá vším, co je na místě pro zajišťování kvality softwaru dělat.

Pojem „péče o kvalitu", „zajištění kvality" a podobně zdomácňuje v naší softwarové tvorbě teprve v posledních létech. Pokud nejen jako pojem, ale i věcně, je to jen dobře. Tedy:

„Péče o kvalitu softwaru je souhrn veškerých opatření a akcí, které jsou nutné k tomu, aby byla potřebná důvěra, že softwarový produkt či jeho komponenta splňuje dané technické požadavky". Tato formulace je asi ne zcela doslovným, avšak výstižným překladem z normy [2]. Jde o činnost, která je neodmyslitelnou součástí řízení softwarového projektu. I zadavatel by při tom měl spolupracovat, protože kvalita softwaru ve smyslu výše uvedeného je (nebo by bytostně měla být) jeho zájmem a rozumné je, aby pro ni něco udělal dříve, než je mu dodán produkt, který řešitel považuje za hotový ale zadavatel ho shledává nepoužitelným.

2.2.2 I kvalita má svoji administrativu

Pro péči o kvalitu softwaru se zpracovává plán (Plán kvality, SQAP). Aby splnil svůj účel, musí jeho věcný obsah být smysluplný a konkrétní, s ohledem na softwarový projekt, kterého se týká. Musí být aktualizován vždy, když život přinesl změnu a (tuzemče, div se) musí být i plněn, což ostatně platí o každém smysluplném dokumentu.

Co tedy závěrem? Pracovně (při vývoji) je lépe považovat kvalitu softwaru spíše za disciplínu, než za kýženou vlastnost, už také proto, že intuitivním kýžením se kvalita zajistí méně spolehlivě, než systematickou péčí o kvalitu. Získávané informace ale nasvědčují tomu, že nejenom svět, ale i některé tuzemské sféry disciplínu péče o kvalitu začínají brát.

2.3 Nepodceňujte počáteční etapy životního cyklu softwaru.

Již v příspěvku [1] byly citovány etapy životního cyklu softwaru tak, jak je uvádějí normy, a s drobnými obměnami s nimi pracují i mnohé další materiály a tvůrčí pracoviště: koncipování, specifikace, návrhu, kódování, testování, zavádění, zkušebního provozu, využívání a údržby, vyřazení z provozu. Podstatou úspěchu není přimět sebe nebo někoho jiného, aby s těmito pojmy pracoval, ale dosáhnout toho, aby nebyly zanedbány relevantní činnosti, které do příslušného období v posloupnosti prací na tvorbě softwaru patří a aby byly provedeny věcně správně a dostatečně podrobně. Za adekvátní podrobnost je účelné považovat takovou, která neumožní přijmout nesprávné nebo neúčelné pracovní rozhodnutí, které by pocházelo z neúplné, nebo nepřesné vstupní informace, která byla v době rozhodování k dispozici.

Vzhledem k pracovní metodice obvyklé u nemála našich pracovníků se jeví důležitějším upozornit na tento závěr důrazněji v našich zemích, než v zahraničí. Tvůrčí pracovníci (a to i zdatní), kteří se dají do konkrétní práce, aniž si řádně ujasní, co se má dokázat, jsou přece v našich zemích dobře známi. Z mnoha příznaků lze usuzovat, že v zahraničí mají takoví pracovníci v průměru menší šanci na úspěch a uznání svých kvalit, než u nás. Doporučení: „Napište to správně hned napoprvé" má přece jen něco do sebe.

2.3.1 Koncipujte řádně problém, který se má řešit

Je známo, že chybné a chybějící závěry počátečních etap se nemusí projevit okamžitě a nemusí být zásadní překážkou v další práci. Jsou však zdrojem chyb, nebo alespoň příležitostí ke vzniku hůře použitelných řešení. Na základě chybného produktu se pak budují produkty další, vytvářením buď vyšších celků jejichž součástí se chybná složka stává, nebo vazeb s tímto chybným produktem.

Je patrno, že čím později, tím bude náprava nedostatků dražší a to spíše řádově, než jen o procenta. Doporučení „Objevte chybu co nejdříve, odstraňte ji tak rychle, jak to jde" z tohoto pohledu v sobě skrývá určitou moudrost.

2.3.2 Trvejte na řádné specifikaci požadavků na software

Požadavek na řádnou specifikaci je přirozený v mnoha oblastech, kde se vytváří něco na míru. Tvorba softwaru takovou oblastí nepochybně je; existující software se málokdy vytváří znovu. Pro znalého tuzemského softwarového pracovníka však může být obtížné zbavit se domněnky, že právě v této oblasti se u nás řádná specifikace tak trochu ignoruje.

Normy ANSI/IEEE (zejména [3]) předpokládají existenci specifikace požadavků na software v písemné formě.

I řádně sestavená specifikace však může doznat změn v průběhu vývoje a jako každý dokument, který má mít smysl, musí být při změně reality vždy aktualizována.

Od specifikace norma výslovně vyžaduje, aby:

(1) byla nedvojsmyslná, t.j. umožnila jen jediný výklad všeho, co se v ní uvádí,

(2) byla úplná, t.j. dala odpověď na každou smysluplnou otázku, kterou si návrhář položí,

(3) její splnění bylo ověřitelné (při testování, předávání a pod.), t.j. použitá formulace musí umožnit zjištění, zda každý jednotlivý bod byl, či nebyl splněn,

(4) byla bezrozporná, t.j. její body si nesmí navzájem odporovat,

(5) byla modifikovatelná, t.j. musí být vytvořena takovým způsobem, že modifikace dílčího požadavku nevede k těžko odhalitelnému narušení některé z požadovaných vlastností,

(6) všechny požadavky v ní byly sledovatelné a to:

(a) zpětně, t.j. musí být zřejmé, kde je jejich původ,

(b) dopředně, t.j. každý požadavek musí být jednoznačně označen, či pojmenován, aby se dalo zjistit, kde všude se vyskytuje a jak se tam projeví.

(7) měla takový obsah a formu, aby byla použitelná i při pracích, které bude nutno provést po ukončení vývoje, t.j. v etapě využívání a údržby softwarového produktu

S uvedenými body je jistě možno souhlasit. Ale specifikaci sestavují lidé, kteří při tom šetří časem, množí nespecifikují rádi a (budeme–li upřímní,) nemusí ani umět dobře formulovat. Proto pozor na záludnosti běžného vyjadřování ! Jen krátká ukázka toho, jak běžně vyjadřujeme co míníme, ale jako specifikace to nevyhoví (např. z hlediska bodu (3)):

- „Program musí fungovat správně", „Produkt musí mít pro člověka vhodnou komunikaci" – Jak exaktně rozhodnout co je „správně", nebo „pro člověka vhodné", až půjde o zaplacení či nezaplacení statisíců a řešitel, který zabudoval do základů programu nešťastně zvolený princip, bude mít na věc jiný názor než zadavatel, při čemž specifikace připouští oba výklady?
- „Program se nesmí dostat do nekonečného cyklu" – sice víme, co se nechce, ale to je požadavek, jehož algoritmickou ověřitelnost popírá sama teorie.
- „Program musí být navržen optimálně" – netřeba komentovat.

Formulace typu „bude stanoveno později" je ve specifikaci nežádoucí. (Je možno zvážit účelnost a vhodnost české zkratky „BSP", která je t.č. volná, místo anglické „TBD" – „to be determined" z normy [3].) Pokud to ale realita opravdu vyžaduje (a to se stává i v zahraničí), norma požaduje: uvést konkrétní důvod, kdy (termínově nebo situačně) „bude stanoveno", opatření jak se „TBD" bude překonávat do té doby a co vše je nutno udělat pro zahlazení „TéBéDéčka", až opravdu „bude stanoveno".

Zadavatelé vývoje, nezapomeňte ani na přijatelnost komunikace software – člověk a na ošetření mimořádných situací; pro některé dodavatele to nemusí být samozřejmou věcí profesionální cti a je třeba to specifikovat podle diskutovaných pravidel !

2.3.3 Dbejte na řádnou analýzu problému

Nedostatečná, stejně jako nesprávná analýza problému může být zdrojem nesprávných pracovních rozhodnutí při návrhu programu (nesprávných ve skutečnosti, ale díky nedostatkům v analýze nemusí být možné tuto nesprávnost objevit už při přijímání rozhodnutí). Nesprávné rozhodnutí má charakter chyby a to chyby spíše principiální, než drobné, která

se opraví jen v místě, kde se nachází. Je typické, že škoda, kterou chyba napáchá, časem roste. Podle analýz různých softwarových firem je tento růst řádový. Odstranění chyby, která unikne i ladění a testování a je objevena až v etapě zkušebního provozu nebo využívání, je zpravidla o několik řádů dražší, než když je tvůrcem zjištěna a odstraněna dříve, než opustí jeho pracoviště.

Zdržte se tedy programování, když ještě nevíte dost přesně, co se má naprogramovat. Pomůže vám to dodržet doporučení: „Napište to správně hned napoprvé".

2.4 Pečujte průběžně o konfiguraci softwaru

Ani pojem „konfigurace softwaru" [4] není u nás důvěrně znám a nebývá považován za samostatnou disciplínu. O co v ní jde ? Softwarový produkt se postupně vyvíjí a už v etapě, kdy není zcela hotový, může být předáván jiným složkám, např. k testování, či k ověřování spolupráce modulů. Při tom je nutné produkt jednoznačně odlišit nejenom od jiných produktů, ale i od všech odlišných provedení téhož produktu. Při tom je lhostejné, zda odlišná provedení vznikla prostým postupným dokončováním vývoje, nebo zda šlo o odstranění chyb, či zda jsou to různé varianty s lišícími se vlastnostmi (např. pro různé uživatele), které mají i v etapě využívání existovat vedle sebe. Bylo by značně neprozíravé postupovat živelně a mít k dispozici vždy jen poslední aktuální vývojovou variantu produktu a nevědět např. které varianty byly komu dodány.

Do péče o konfiguraci tedy patří souhrn těch činností či služeb, které zajišťují aby jednotlivé vyvinuté varianty komponent softwaru, které byly někam s nějakým cílem předány (k užívání, k zařazení do vyššího celku, k otestování a j.), byly také jednoznačně označeny a aby vždy bylo možno se na ně odkazovat a vracet se k nim bez nebezpečí, že budou zaměněny s jinými variantami téhož produktu. Totéž musí péče o konfiguraci zajistit i pro všechny verze veškeré dokumentace, pro testovací či jiná data a pro všechny ostatní materiály, které s vyvíjeným softwarem souvisejí. Musí být zajištěna i jednoznačnost všech křížových vazeb, t.j. např. pro každou konkrétní verzi softwarového produktu musí být možno stanovit, které verze dokumentace pro ni platí, ze kterých variant složek jsou různé varianty vyššího celku sestaveny, atd. V této šíři už není péče o konfiguraci tak snadnou záležitostí.

Zcela nepochybně do péče o konfiguraci patří identifikační schéma pro jednotlivé složky projektu, které musí zajistit dostatečně přesné a trvalé označení jejich jednotlivých vývojových variant. Dále pak i metodika přidělování jednotlivých označení a registrace jejich přidělení.

2.5 Vytvářejte adekvátní dokumentaci

„Domácí softwarovi tvůrci mají všeobecný odpor k administrativě". V rámci toho neradi píší i dokumentaci. Raději argumentují tím, že „moderní programovací nástroje vytvářejí dokumentaci automaticky a tedy bez chyb a vždy aktuální", nebo že program je

vybaven nápovědnými texty, které dokumentaci nahrazují. Možná by užasli, kdyby si přečetli něco z toho, co praví normy o dokumentaci, ale k tomu nebývají ochotni. Stačilo by, kdyby si prohlédli uživatelskou dokumentaci některého produktu světové softwarové firmy, kterou uznávají. Není třeba se obávat, že by taková dokumentace neexistovala. Nesměšujme to však s případem, kdy ji tito pracovníci nemají, možná proto, že software získali nestandardní cestou.

Uživatelskou dokumentaci vytvářejme se snahou o dokonalost, a to z hlediska toho uživatele, kterému software slouží a nikoliv z hlediska parametrů osoby jejího tvůrce. Norma [5] zde nabádá nejen aby existovala dokumentace referenční i popisná, ale aby případně existovala různě zaměřená dokumentace pro různé okruhy uživatelů, vždy každému okruhu „na míru". Vlastní software (pro potřebu budoucí údržby a změn) pak dokumentujte s důrazem na správnost a úplnost popisu produktu. Využití různých dokumentujících výpisů je přínosem, ale předem tvrdit že to jako jediné postačí, je lépe považovat za unáhlené.

Dokumentace má být k dispozici současně se softwarem; kdyby to bylo ještě dříve, nebylo by to na škodu.

2.6 Dejte prostor pro testování a ladění a rozlišujte tyto procesy

Oba pojmy sice používáme, ale máme sklony je nerozlišovat. Možná jsme zvyklí na prostředí, kde obě aktivity provádí tentýž pracovník a stálým opakováním procesu za stejných podmínek se nutnost rozlišení ztrácí. Profesní kolegové ze světa však takové nerozlišování ne zcela snadno chápou, pro ně jsou to dva formálně zcela odlišné procesy.

Ladění je totiž součástí činnosti tvůrce kódu programu, jímž si ověřuje, zda navržený kód programu je správně zapsán a zda má ten efekt, který tvůrce zamýšlel. Ladění zahrnuje jak zjišťování chyb, tak jejich opravu a to v krátké zpětnovazební smyčce, bez formalit. Výsledkem ukončeného ladění je softwarový produkt, o němž je jeho tvůrce přesvědčen, že funguje tak, jak zamýšlel. Ladění z tohoto pohledu představuje nejnižší (možná je přijatelné říci „fyzickou") úroveň procesu, který zajišťuje správnost funkce některé softwarové položky.

Naproti tomu testování je proces, který má charakter kontroly. Typickou pracovní otázkou pro testování není „zda je to napsáno správně", ale spíše „zda je splněno to, co bylo požadováno". Vlastní náprava zjištěných nedostatků není předmětem testování, produkt se vrací řešiteli jako nevyhovující s tím, že až ji řešitel opraví (a odladí), testování se musí opakovat. Cíle obou procesů nejsou samozřejmě v protikladu, ale pohled na produkt je při testování veden z hlediska zadavatele, či uživatele a pro korektní, neintuitivní hodnocení jsou zapotřebí specifikace, tedy informace co se požadovalo a čeho mělo být při tom dosaženo.

Zajistěte tedy i řádné otestování všech složek vytvářených softwarových produktů (tedy mimo jiné i dokumentace). Chyby objevované až při využívání jsou výrazně špatnou známkou pro tvůrce (pracoviště, kolektiv, firmu, společnost, či co jste). Připusťte si tuto skutečnost, i když se zdá, že na ní stále ještě nezáleží tolik, jako v často citovaném zahraničí. Jde-li náhodou o software z kategorie „kritický", je hledání chyb až v etapě využívání už z principu dost nereálné. Vlastnosti výsledného produktu budou vždy neformálním ohodnocením (ale mohou být i znehodnocením) vašeho jména.

Literatura

[1] Šlegr J.: Programová tvorba z pohledu světových norem, seminář Programování '92, Ostrava.

[2] ANSI/IEEE Std 983–1986, IEEE Guide for Software Quality Assurance Planning

[3] ANSI/IEEE Std 830–1984, IEEE Guide to Software Requirements Specifications.

[4] ANSI/IEEE Std 828–1983, IEEE Standard for Software Configuration Management Plans.

[5] ANSI/IEEE Std 1063–1987, IEEE Standard for Software User Documentation.

Autor: Ing. Jiří Šlegr CSc, SPT Telecom – IAT Praha,
Holečkova 10, 125 07 Praha 5,
tel. (02) 54 34 51, linka 244.