Má také u softwaru smysl mluvit o "konfiguraci" ?

Jiří Šlegr

Motto:

"SCM is a Service Function."
"SCM is a Part of the Engineering Process."
"SCM Manages all Software Entities."
ANSI/IEEE Std 1042-1987

1. Něco úvodem

V loňském [2] a předloňském [1] příspěvku byl podán nástin toho, jak si standardy ANSI/IEEE představují metodiku tvorby softwarových produktů a bylo formulováno určité poučení z vývoje (z pohledu těchto standardů do jisté míry asi krizového) v našich krajích v této oblasti. Letošní příspěvek si vzal za své jeden dílčí námět: jak nahlížet na pojem "konfigurace softwaru", který se v uvedených standardech na dosti místech používá. Na obě slova jsme sice zvyklí, nicméně slýcháme, či používáme je spíš odděleně. Pojem "konfigurace softwaru" není u nás důvěrně znám a nebývá považován za samostatnou disciplínu. O co tedy jde ?

2. Konfigurace softwaru tedy ano, či ne ?

Pro začátek se na věc podívejme tak, jako kdyby "konfigurace" znamenala u softwaru totéž, co u hardwaru. Tak jako sestava hardwaru nějakého zařízení (v prostředí pro nás obvyklém jsme zvyklí na počítače) se skládá z dílčích částí (zde většinou zařízení), nelze popřít, že i u softwarového produktu je tomu obdobně. A pokud nejsme konzervativní, pak připustíme (a to při vší úctě k hardwaru), že problematika s takovou "konfigurací" může být u softwaru přece jen poněkud komplikovanější, nebo snad přesněji vyjádřeno: skrytější. Realita existence různých dílčích složek softwaru může snáze uniknout pozornosti uživatelů či provozovatelů, než je to u součástí hardwaru, které jsou více fyzické, či hmotné a představa, že taková složka chybí, nebo že to není ta správná, má proto konkrétnější podobu, než u softwarových entit.

Ve světě je běžně znám pojem "[software] configuration management" i tomu odpovídající zkratka SCM, česky bychom to možná nazvali "správa konfigurace [softwaru]". Je dostatek důvodů k tomu, abychom něco takového považovali za samostatnou disciplínu ? Pokud mechanicky vezmeme na pomoc světoznámé standardy, pak nepochybně musíme odpovědět, že ano. Vždyť např. příručka pro správu konfigurace (tak by se jeden z citovaných standardů ANSI/IEEE asi jmenoval) je několikrát tlustší, než např. standard z téže řady pro uživatelskou dokumentaci. Lépe však bude probrat, co vše je oním pojmem míněno, či co je účelné, aby jím míněno bylo. Začít můžeme v prostředí nám známém.

Pokud jsme v loňském příspěvku [2] přijali za rozumný výrok (B): "funguje systém tvorby programů" , pak nás to nabádá k odpovědi "ano" na otázku položenou v nadpisu této kapitoly (šlo

tehdy o srovnání dvou charakteristických modelových systémů tvorby softwaru, takže výrok (B) stál proti výroku (A), který zněl: "fungují zdatní jedinci").

Aniž bychom zabíhali do podrobností, je zřejmé, že všude na světě se softwarový produkt vyvíjí postupnými kroky, po částech a jednotlivé složky velkých celků často na více pracovištích paralelně. K tomu je nutná určitá (a výstižnější je říci: dobrá) koordinace, aby vše probíhalo účelně z hlediska dosažení cíle, kterým je nepochybně vytvoření softwarového produktu. Jistě nebude mít charakter pomluvy vlastní tvůrčí dílny ani vyjádření, že u rozsáhlého softwarového produktu nastane někdy situace, kdy ani odpovědní pracovníci vývoje softwaru (pokud existují) nejsou schopni zajistit odpovědi na konkrétní otázky typu "jak přesně vypadá verze produktu, která byla dodána společnosti PARKINSON a.s. ?", "otestoval někdo modul BLUFF.FUJ a s jakými výsledky ?", či "platí ještě dokumentace, která byla vydána asi před rokem, vlastní ji náš šéf a také se pozná podle toho, že má na přední straně ten zajímavý znak ?". A to jsme neuvažovali situaci, že by ministerstvo účtování změnilo pravidla pro výpočet odvodu z úbytě, což se čas od času stává i v docela spořádaných pospolitostech. Problémy a potřeba je řešit tedy vznikají jak z nedokonalostí samého procesu tvorby softwaru, tak z reality životního dění v pospolitosti a ignorovat je by asi nebylo o mnoho rozumnější, než zrušit úrazovou službu s předsevzetím, že nadále budeme všichni velice opatrní.

3. Správa konfigurace

Pokud jsme přijali výše zmíněný výrok (B), pak správa konfigurace bude něčím, co po věcné stránce napomáhá pravdivosti tohoto výroku. Ovšem pokud bychom byli zůstali při výroku (A), bylo by větší nebezpečí, že pole působnosti bude mít modelová situace, popsaná asi o odstavec výše, v charakteristice vlastní tvůrčí dílny.

Pokud někdo váhá čeho se stoupenci správy konfigurace softwaru obávají, ať si zkusí představit větší, na zakázku vyvinutý softwarový systém, který má desítky dílčích celků, byl implementován ve funkčně ne zcela identických variantách na desítkách pracovišť a po čase z některého přijde reklamace, či nový požadavek. Aby se předešlo situaci, že dodavateli softwaru (míněno "odpovědnému", protože takový by měl být každý dodavatel) vlastně nebude známo, jak přesně vypadá a z čeho se skládá systém, ke kterému uživatel projevil výhrady či požadavky, musí slušně fungovat něco, co bychom mohli právě nazvat "správa konfigurace". A nepostačí, aby zahrnovala jenom dostatečně "chytře" navržený číslovací plán položek softwarových produktů (o něm zatím nebyla řeč), ale hodí se i jakési "zafixované" provedení softwarového produktu a jeho komponent a při nejmenším bude nutná i zachovaná informace o realizovaných dodávkách.

Jaké si tedy vzít ponaučení ? Neklaďme si už raději otázku z nadpisu kapitoly 2 a všimněme si, jaké základní funkce správa konfigurace má. Není při tom třeba odvodit vše od začátku, vždyť už se stalo. Standardy ANSI/IEEE uvádějí několik základních funkcí správy konfigurace softwaru: **stanovení konfigurace, správa fixovaných verzí, řízení změnového procesu a správa knihoven, správa informace o konfiguraci, přezkoumávání a prověrky, proces vydávání.** Vzhledem k původu je to jistě výčet kvalifikovaný a je tedy možno se podívat, co podle těchto pramenů do uvedených činností patří.

3.1. Stanovení konfigurace

Sem je možno zahrnout počáteční činnosti, které mají co dělat se správou konfigurace. Je třeba dát k dispozici identifikační schéma, které dokáže obsáhnout celou šíři struktury produktu (t.j.

jednoznačně v ní označit každou existující položku) a je žádoucí projevit při tom maximum předvídavosti, aby si schéma tuto schopnost zachovalo po celou dobu života vyvíjeného softwaru, aniž by muselo být měněno, t.j. aby nedorazilo na meze svých možností. To není snadné, protože v době, kdy se identifikační schéma vytváří, je struktura budoucího produktu zřídkakdy známa do takových podrobností, s jakými bude muset schéma později pracovat.

Správa konfigurace se však vztahuje na všechny druhy softwarových entit, ze kterých se vyvíjený softwarový produkt skládá, nebo které s jeho vývojem nějak souvisejí. Potenciálně tedy jde o:

- Specifikace (např. požadavků na vývoj softwaru)
- Zdrojový kód programu (na počítačově čitelných médiích)
- Relativní a proveditelný strojový kód programu
- Softwarové knihovny a databáze
- Návrh testů, specifikace pro testovací příklady a procedury
- Testovací data a procedury pro jejich generování
- Uživatelská dokumentace
- Dokumentace pro potřebu údržby (výpisy programu, datové slovníky, materiály z detailního návrhu atd.)
- Pomocný a podpůrný software použitý při vývoji
- Administrativní a plánovací dokumenty softwarového projektu

3.2. Správa fixovaných verzí

S pojmem "fixovaná verze" se autor v domácích textech zatím nesetkal. Pracovně ho použil proto, že bylo třeba nějakého ekvivalentu pro anglický pojem "baseline", kterého jsou leckteré ze zmíněných standardů plny a z různých možností se toto zdálo být v uvedeném významu nejpřijatelnější. Nelze se ubránit dojmu, že je to pojem, bez kterého softwarová tvorba v různých částech světa není schopna pracovat. Podívejme se tedy co to vlastně je, když se to dá dokonce "spravovat".

Není to nic těžko představitelného, jenom snad zatím v češtině standardně nepojmenovaného. Softwarový produkt se musí nejprve vyvinout. Pokud postupujeme rigorózně, pak stručně řečeno, nejprve na základě potřeby vyspecifikujeme, co má být výsledkem vývoje, navrhneme hrubou a detailní strukturu produktu, jednotlivé složky produktu naprogramujeme (pokud jde o dokumentaci, tak tu napíšeme), vše po částech i v celku otestujeme, implementujeme, atd. a když je vše hotovo, pak produkt ještě udržujeme, dokud ho někdo používá. Je zřejmé, že při tom vzniká řada variant produktu už díky tomu, že se každý objekt postupně dotváří, při čemž už i ten ne zcela hotový může být podroben určitému testování, přezkoumávání nebo jiné činnosti mimo stůl vývojáře-autora. Připustíme-li, že v produktu může být i chyba, pak je tu další dimenze, na níž vznikají různé varianty provedení. Je třeba rozlišit všechny takovéto varianty provedení od sebe, nebo vyjádřeno ještě náročněji: zcela jednoznačně identifikovat jakoukoliv existující konkrétní variantu a to kdekoliv a kdykoliv, při čemž čtenáře jistě není třeba přesvědčovat, že pro úspěšnost procesu vývoje je to důležitější, než vést statistiku, jak měl kdy který programátor úhledně vyplněný pracovní výkaz.

Avšak nejen že fixovaná verze každé položky konfigurace softwarového produktu musí mít přiřazen konkrétní a evidovaný vyhrazený identifikátor, je třeba, aby zdrojový program, relativní modul, proveditelný program, dokumentace a nástroje pro podporu vývoje a údržby softwarového produktu, byly zachyceny a uchovány a nějakým způsobem pojímány či registrovány jako součást určité fixované verze. Jak se počítačový program postupně vyvíjí od počátečního záměru na jeho vytvoření, až do etapy údržby hotového produktu, vytváří se tím řada stále kompletnějších

fixovaných vývojových verzí. Abychom viděli, jak je "fixovaná verze" pro správu konfigurace užitečná, připomeňme některé její přínosy:

- Aktuální stav vyvíjeného produktu je vždy dán poslední vytvořenou fixovanou verzí, do které jsou vneseny všechny následně provedené změny,
- Umožňuje navzájem rozlišit různé interní verze, které jsou dodávány zákazníkům stále pod tímtéž globálním názvem (t.j. rozlišují se postupně dodávaná avšak v podrobnostech se lišící provedení téhož produktu jako celku),
- Umožňuje evidovat jednotlivá interní vydání položek konfigurace při jejich předávání mezi etapami vývoje a k testování,
- Je pomůckou při zajišťování úplnosti a aktuálnosti dokumentace produktu,
- Pomáhá uplatňovat standardy péče o kvalitu softwaru,
- Umožňuje evidovat případné existující vazby zákazníka na vnitřní (vývojové) verze a vycházet ze správných předpokladů při řešení problematiky konkrétních uživatelů (požadavky, změny).

3.3. Řízení změnového procesu

Provádění změn v softwarovém produktu je zvnějšku dobře viditelným a relativně snadno pochopitelným procesem a to ať je jeho důvodem odstranění chyb objevených až během užívání softwaru, nebo respektování změn, které přinesl reálný život; na obojím má totiž koncový uživatel zájem. Při tom požadavek na změnu může přijít jak od uživatele (který softwarový produkt již používá), tak z vývojového pracoviště.

O tom, že bude provedena změna však nemůže rozhodovat kdokoliv, i to by u trochu rozsáhlejšího softwarového projektu postupnými kroky a systematicky zavedlo chaos a dala by se najít analogická modelová situace, jako v kapitole 2. To však uctívané standardy vědí a proto ke změnovém procesu mnohé uvádějí.

Změny se provádějí formálně, nikoliv živelně. O změnu se žádá a žádost prochází schvalováním. Příslušný kompetentní orgán je předem stanoven a nemusí to být tentýž pro změnu čehokoliv a kdykoliv. Ve standardech existuje vžitý pojem: "Rada správy konfigurace" (Configuration control board, CCB), jejíž jmenování se v každém softwarovém projektu předpokládá a je to orgán, kterému byla dána řídicí pravomoc posuzovat a na základě výsledku posouzení schvalovat či neschvalovat navržené technické, logické, či programátorské změny, a zajišťovat pak jejich řádné provedení. CCB se při tom více zaměřuje na manažerské (co vše to bude představovat), ekonomické (náklady na provedení změny), legislativní (dodržení obecně platných předpisů a standardů), smluvní (vliv na splnění kontraktačních povinností a termínů) a podobné aspekty, než na ty ryze "softwarově odborné".

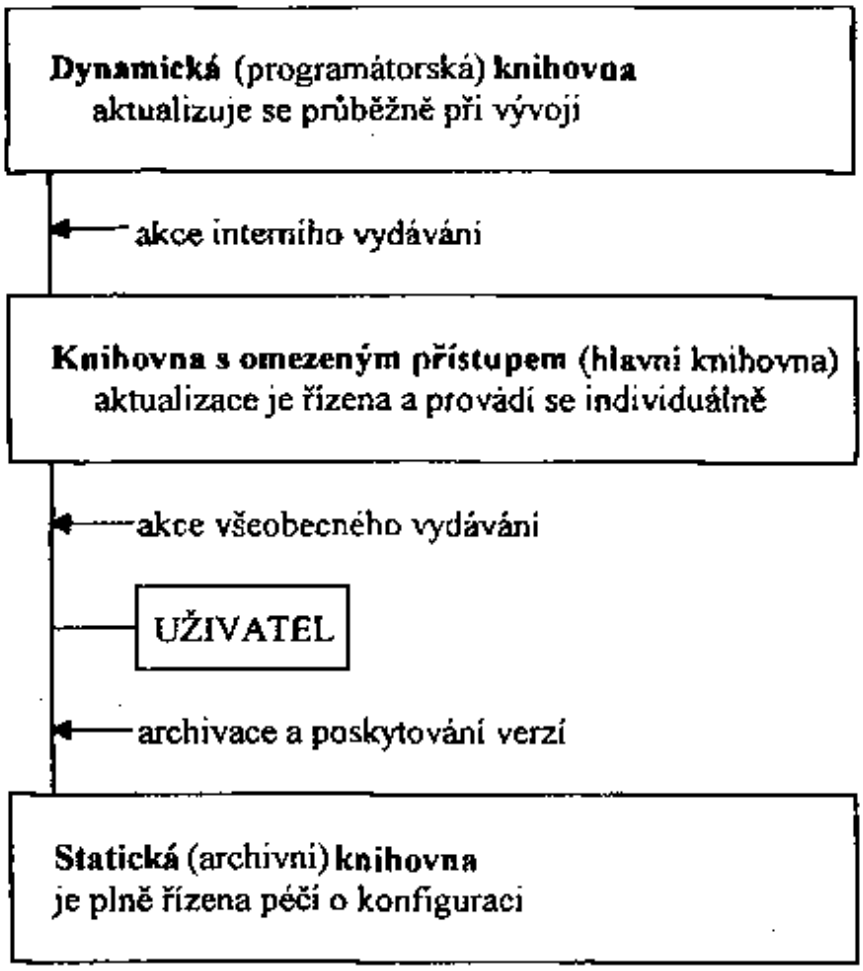
3.4. Správa knihoven

Správa softwarových knihoven je samostatnou funkcí, avšak se změnovým procesem ji spojuje skutečnost, že většina změn se přímo týká softwaru. Počet a druh použitých knihoven se může lišit od projektu k projektu, ale z hlediska správy konfigurace se principiálně pracuje s třemi druhy knihoven, jak je to znázorněno na obr. 1.

Dynamická či programátorská knihovna slouží pro uchování nově vytvořených, nebo modifikovaných softwarových celků (bloků, modulů, datových souborů a s nimi související dokumentace). Programátoři sem průběžně ukládají všechny informační celky v takové formě, v jaké s nimi při vývoji pracují.

Knihovna s omezeným přístupem či **hlavní knihovna** se používá pro správu užívaných fixovaných verzí a pro jejich změnový proces. Zde se udržují položky konfigurace, které byly vnitřně vydány pro potřebu integrace do vyšších celků. Změny položek uložených v této knihovně, musí být povoleny a schváleny (viz 3.3) a zápis nových objektů do této knihovny je řízen. Čtení však není striktně omezeno.

Statická či archivní knihovna se používá pro archivaci různých fixovaných verzí, které byly v průběhu vývoje vydány pro externí užívání. Zde se udržují vzory a ostatní autorizované verze položek konfigurací počítačových programů, které byly vydány pro externí užívání.



Obr. 1. Druhy softwarových knihoven z hlediska správy konfigurace

3.5. Správa informace o konfiguraci

Tato činnost sice netvoří tak vyhraněnou kategorii, ale mluví se o ní samostatně proto, že je třeba aby bylo včas jasno, jaká informace se má zachycovat a uchovávat a dále pak v jaké formě, formulaci a kdy a komu ji bude třeba poskytovat. Nejde jen o informaci prvotně vznikající a očekávanou, ale i o tu nepříliš očekávanou, jako jsou např. hlášení o zjištěných chybách a problémových stavech. Je-li projekt dobře organizován, pak musí také **fungovat jako systém** a standardně musí proběhnout i smysluplné zpracování informace takovéhoto druhu.

3.6. Přezkoumávání a prověrky, proces vydávání

Do této funkce patří kontrolní procesy, které souvisejí s konfigurací softwaru. Typicky je to fyzická prověrka, t.j. zjišťování úplnosti a správnosti softwarové konfigurace a funkční prověrka, t.j. zjišťování zda konfigurace funguje podle specifikací, přesněji řečeno zda všechny položky byly otestovány předepsanými postupy a zda výsledek testování byl vyhovující ve smyslu daných kritérií.

Proces vydávání musí být naplánován a probíhat tak, aby příjemce (t.j. uživatel) měl dostatek informací a materiálů potřebných k užívání softwarového produktu (z hlediska konfigurace softwaru to představuje seznamy dodávek softwaru, instalační a provozní pokyny, manuály, nároky na (hardwarové a softwarové) prostředí a pod.

4. I konfigurace má svoji administrativu

V rámci správy konfigurace softwaru se zpracovává plán (Plán správy konfigurace, SCMP). Pro usnadnění orientace v různých plánech vyžaduje standard [4] pevnou osnovu SCMP (která se nemá narušit ani v případě irelevantnosti některých bodů), stručně takto:

1. Úvod (účel, zaměření, definice, odkazy)
2. Řízení
2.1. Organizace
2.2. Pověření činnostmi SCM
2.3. Správa rozhraní
2.4. Realizace Plánu SCM
2.5. Koncepční záměry, směrnice a postupy
3. Náplň činností SCM
3.1. Stanovení konfigurace (identifikace fixovaných verzí, pojmenovávání, označování a číslování)
3.2. Operativní řízení konfigurace (Pravomoc pro povolení změny, změnová řízení, rada správy konfigurace (CCB), podpůrný software)
3.3. Správa informace o konfiguraci
3.4. Prověrky a přezkoumávání
3.5. Proces vydávání
4. Nástroje, techniky a metodiky
5. Správa externích dodávek (software od subdodavatelů a prodejců)
6. Shromažďování a uchovávání záznamů

5. Závěr

Přesvědčit zdatného jedince, aby prosazoval správu konfigurace jako disciplínu, s cílem aby lépe fungoval systém, nebývá snadné. Pokud tento příspěvek udělal aspoň trochu jasno v názorech čtenářů na to, co je "konfigurace" v případě softwaru, pak splnil svůj úkol.

Literatura :

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

[2] Šlegr J., Na co by si měli naši tvůrci softwaru zvyknout, seminář Programování '93, Ostrava.

[3] ANSI/IEEE Std 1042-1987, IEEE Guide to Software Configuration Management.

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

Autor :

Ing. Jiří Šlegr CSc, SPT Telecom, a.s. Praha,
Bassova 14, 190 00 Praha 9,
tel. (02) 66 31 30 01, linka 304.