Programová tvorba z pohledu světových norem

Jiří Šlegr

1. Úvod

Příspěvek je pokusem autora sdělit informace získané z některých norem ANSI/IEEE, které se týkají programové tvorby a s kterými měl možnost nějaký čas se zabývat. Vzhledem k tomu, že pramenem informace je materiál takového charakteru, jakým jsou normy, při čemž celkový rozsah tohoto zdroje je mnohonásobně větší než rozsah tohoto příspěvku, je reálné zde uvést jen hlavní ideje a postřehy, zajímavé pro tuzemského tvůrce programů.

Vytvoření tohoto příspěvku bylo spojeno i s některými jazykovými problémy, protože v textu norem jsou používány i pojmy, které nemají v češtině dostatečně přesný a zároveň používaný ekvivalent. Někdy dokonce ani reality, které jsou těmito pojmy označovány, nemají v čs. prostředí ekvivalent (např. když se několik činností navzájem nerozlišuje, protože je zvykem je chápat jako jeden celek). V takových případech byly použity existující české pojmy, které se zdály být nejpřijatelnější. Někde jsou v textu připojeny i užívané anglické termíny a kde jsou všeobecně užívané, také jejich zkratky. Hlavním cílem příspěvku však zůstalo vysvětlení obsahu původního pramene. Příspěvek si všímá těchto témat:

(1) V jakém prostředí a podmínkách pro programovou tvorbu zmíněné normy vznikaly,

(2) Jaký celkový metodický přístup k programové tvorbě tyto normy předpokládají,

(3) Etapy životního cyklu softwaru,

(4) Činnosti v rámci programové tvorby a jaké dokumenty či dokumentace se při vývoji programů vedou a používají.

2. Vliv prostředí a podmínek na softwarovou tvorbu

Je zřejmé, že obecně použitelná metodika musí být taková, aby vyhověla vždy, tedy aby neselhala ani za okolností, které jsou z nějakého hlediska obtížné či zvláštní. Je to zejména:

(1) Kritičnost aplikace:

Metodika musí pamatovat i na specifika kritických aplikací, které na př. řídí reálné procesy, uchovávají a poskytují závažná data, či provádějí finanční transakce, takže chyby a havárie by měly dopady na realitu, čímž i následky by mohly být katastrofální. Software

zde musí mít takové vlastnosti, aby sám významně nepřispěl ke zvýšení pravděpodobnosti výskytu takových havarijních situací nad míru, která je dána jinými činiteli.

(2) **Rozsah projektu:**

Metodika musí vyhovět i v případě, že jde o velmi rozsáhlé softwarové projekty, jejichž vytvoření je nad síly jednotlivce, či několika vybraných tvůrčích osobností. Uvést do provozu bez chyb velmi rozsáhlý softwarový projekt vyžaduje korektní a účelný postup, který musí být realizovatelný s takovými pracovníky, jací jsou k dispozici. K tomu musí být zvoleny adekvátní postupy a musí řádně fungovat řízení a koordinace prací.

(3) **Záruka dokončení**

Vývoj softwaru byl zadán proto, že jeho budoucí uživatel je přesvědčen, že ho ke své činnosti bude potřebovat a je tedy ochoten za jeho vývoj zaplatit. Je však ochoten objednat jeho vývoj jen v případě, že nemá pochyb o tom, že výsledkem vývoje bude použitelný softwarový produkt.

(4) **Průběžná doba realizace**

Doba vývoje složitého produktu, jakým software je, nebývá příliš krátká a proto spolu s předchozí podmínkou rozhoduje i termín, kdy bude požadovaný software k dispozici. Krátké dodací termíny jsou tedy často významným činitelem při rozhodování o volbě dodavatele.

(5) **Minimalizace celkových nákladů**

Cena softwaru musí uhradit náklady na vývoj a být zdrojem určitého zisku. Neúčelně vynaložené a tím zbytečně vysoké náklady ovlivní nepříznivě buď zisk, nebo cenu a v tom případě možná i rozhodnutí o volbě dodavatele. Je známo, že velkou část celkových nákladů na software nespotřebuje jeho prvotní vývoj, ale pozdější údržba softwaru. Náklady na nutnou údržbu budou klesat s počtem chyb, které zůstanou v předaném softwaru, i s obtížností provádění dodatečných změn, které se i v bezchybném softwaru ukáží jako nutné z jiných důvodů. Malý počet neobjevených chyb, stejně tak jako relativní snadnost provádění změn, je právě produktem dobré metodiky návrhu a dobré dokumentace programů.

(6) **Kvalifikace a odborná úroveň pracovníků:**

Vývoj softwaru musí používat účelnou dělbu práce, která využije ty pracovníky, kteří jsou k dispozici, nebo ty které je možno získat a také je v průběhu prací bez vážnějších potíží nahradit jinými. Rozdělení úkolů a odpovědností v rámci dělby práce a vytvoření funkční hierarchie musí být konzistentní, aby jeho nedostatky nebylo nutno vyvažovat jinak neodůvodněným zvýšením požadavků na kvalifikaci.

Není prozíravé zakládat realizaci projektu na jednom nebo několika konkrétních, byť vysoce schopných a kvalifikovaných jedincích, kteří jsou obtížně nahraditelní. Mohou sice být přínosem pro rychlý postup prací zejména v počátečních fázích, ale zárukou

úspěšného dokončení projektu jsou už mnohem méně. Stejně tak netrvat na řádném zadokumentování softwaru a spoléhat se na využití kvalit takových jedinců při budoucí údržbě, je stejným hazardem.

3. Rysy výsledné metodiky

Metodika programové tvorby podle těchto norem, se liší v mnoha rysech od metodiky, na kterou jsme na našich domácích pracovištích zvyklí. Charakteristické rysy „světové" metodiky lze formulovat takto:

(1) Metodika je propracována do podrobností. Jednotlivé dílčí činnosti jsou dost specializované a je také poměrně přesně specifikováno co do nich patří a co nikoliv. Jednotlivé činnosti se navzájem zřetelně rozlišují.

(2) Programová tvorba má mnohem více charakter systematické rutinní, či průmyslové činnosti, než jsme zvyklí. Nevychází se z principu, že softwarový produkt bude výsledkem intuitivní práce několika erudovaných, odborně zdatných pracovníků, kteří vytvoří vše, co bylo uloženo jim i jiným, a kteří nakonec vytvoří i to, na co se prostě zapomnělo.

(3) Skoro všechny činnosti se podrobně a poměrně rozsáhle dokumentují. Pracuje se s mnoha plány, od kterých se požaduje, aby počítaly se změnami, jejichž potřeba se objeví až v průběhu prací, aby upozorňovaly na možná rizika a aby pro takové případy alespoň rámcově navrhovaly opatření, která budou přijata, jestliže se příslušné riziko ukáže jako reálné. Zcela jednoznačně se však předpokládá, že plán, či dokument je přínosem pro prováděnou činnost. Jedině to je důvodem jeho existence a tomu musí odpovídat i jeho obsah. Ten musí být aktualizován (a to daným postupem), jestliže se změní skutečnost, které se týká.

Mnoho informací se průběžně zaznamenává a archivuje. Účelné používání výpočetní techniky při tom normy velmi doporučují a počítají i se sběrem statistických dat, z jejichž zpracování je možno se poučit a následně zdokonalovat proces návrhu softwaru.

(4) Předpokládá se nutnost dodržování řady norem a předpisů, které musí být v projektu vyjmenovány. Lze přijmout i rozhodnutí, že v určitých konkrétních případech nebudou některé tyto normativní dokumenty respektovány, ale musí to být zdůvodněno. Počítá se s existencí mnoha takových předpisů, jejichž platnost je jen lokální (podnikové, odvětvové, vojenské a pod.).

4. Etapy životního cyklu softwaru

Členění životního cyklu softwaru na dílčí etapy je normalizováno a důsledně užíváno takto (jsou uvedeny odpovídající anglické termíny, i vysvětlení co je jimi míněno):

Životní cyklus softwaru (Software life cycle) : *Období které začíná, když se objeví první zmínky o softwarovém produktu nebo o jeho potřebě a končí když tento produkt přestane být k dispozici pro použití.* Skládá se z těchto etap:

(1) **Etapa koncipování** (Concepts phase): *Období kdy jsou popisovány potřeby budoucího uživatele a jsou dokumentačně zachycovány a vyhodnocovány.*

(2) **Etapa specifikace** (Requirements phase, Development phase) : *Období kdy jsou formulovány a dokumentovány požadavky na softwarový produkt, jako např. jeho požadované funkční a výkonové možnosti a vlastnosti.*

(3) **Etapa návrhu** (Design phase) : *Období kdy se navrhuje a dokumentuje architektura vyvíjeného softwarového produktu, jeho rozdělení na složky, styková rozhraní a formáty dat a ověřuje se, zda tento návrh splňuje specifikované požadavky.*

(4) **Etapa kódování** (Implementation phase) : *Období kdy se softwarový produkt vytváří (programuje se) podle vývojové dokumentace a odlaďuje se.*

(5) **Etapa testování** (Test phase) : *Období kdy se jednotlivé složky softwarového produktu vyhodnocují, zda splňují specifikované požadavky a posléze se spojují ve větší celky a zjišťuje se, zda postupně tyto celky, až nakonec i softwarový produkt jako celek, splňují požadavky, které byly pro něj specifikovány.*

(6) **Etapa zavádění** (Manufacturing phase) : *Období kdy základní verze softwarového produktu je přizpůsobována jednomu vybranému operačnímu prostředí a je předávána na toto pracoviště uživatele.*

(7) **Etapa zkušebního provozu** (Installation and checkout phase) : *Období kdy softwarový produkt je začleňován do svého operačního prostředí a v něm je testován, zda se chová tak, jak bylo specifikováno v požadavcích.*

(8) **Etapa využívání a údržby** (Operation and maintenance phase) : *Období kdy se softwarový produkt využívá v běžném uživatelském prostředí, při tom se sleduje, zda jeho využívání je bezproblémové a modifikuje se s cílem jednak odstranit případné zjištěné problémy či chyby a jednak přizpůsobit ho, aby vyhověl dodatečně změněným, či rozšířeným požadavkům. Je to hlavní etapa životního cyklu.*

(9) **Etapa vyřazení z provozu** (Retirement phase) : *Období v němž přestává být zajišťována dostupnost softwarového produktu pro uživatele*

5. Standardní činnosti v rámci softwarové tvorby

5.1. Taxonomie činností

Činnosti při tvorbě softwaru jsou (zejména v [2]) děleny a charakterizovány takto:

PRACOVNÍ ČINNOST PŘI SOFTWAROVÉ TVORBĚ (Job function) : Skupina tvůrčích procesů která je chápána jako celek pro účely organizace práce, přidělování úkolů či jejich hodnocení. Příkladem je vývoj, testování či péče o konfiguraci.

Tvorba produktu (Product engineering function) :

Analýza požadavků (Requirements Analysis)

Studium potřeb uživatele s formulováním požadavků na software

Návrh (Design)

Vytváření architektury, složek, modulů, rozhraní, testovacích přístupů a dat pro softwarový systém tak, aby byly splněny formulované požadavky

Programování (Coding)

Převod logiky a dat ze specifikací pro návrh do zápisu v programovacím jazyku.

Sestavování (Integration)

Vytváření kompletních systémů ze složek softwaru, hardwaru, nebo z obojího.

Převod (Conversion)

Modifikace existujícího softwaru pro jiné prostředí (jazyk, počítač, ..)

Ladění (Debugging)

Proces vyhledávání, analýzy a oprav chyb, na jejichž existenci je podezření.

Zabezpečení produktu (Product Support)

Poskytování informací, školení a jiné pomoci v souvislosti s instalací a uváděním softwaru do provozu v prostředí, pro které je určen a v souvislosti s distribuováním jeho zdokonalení mezi uživateli.

Údržba softwaru (Software Maintenance)

Modifikace již dodaného softwaru kvůli opravě chyb, dosažení jiných vlastností, nebo přizpůsobení změněnému prostředí.

Verifikace a schvalování (Verification and validation) : Proces který určuje, zda požadavky na systém nebo na jeho složku jsou úplné a správné, zda produkty vzniklé v jednotlivých etapách vývoje splňují požadavky či podmínky dané v předchozí etapě a zda výsledný systém či jeho složka vyhovuje specifikovaným požadavkům.

Ověřování a kontroly (Reviews and audits)

Vyhodnocování složek softwaru či stavu projektu ke zjištění, zda nedochází k odchylkám od plánovaných výsledků a navrhování nápravy. Vyhodnocení zda jsou dodrženy normy, směrnice a specifikace.

Analýza produktu (Product Analysis)

Proces vyhodnocování produktu ručními nebo automatizovanými prostředky, který má určit zda produkt má jisté vlastnosti.

Testování (Testing)

Proces zkušebního používání systému nebo jeho složky, nebo jeho vyhodnocování ručními nebo automatizovanými prostředky, prováděný s cílem ověřit, zda systém splňuje specifikované požadavky, a objevit nesouhlas mezi očekávanými a skutečnými výsledky.

Technické řízení (Technical Management Functions) : Užití technických a administrativních prostředků pro plánování, organizaci a řízení inženýrských činností

Řízení procesu (Process Management)

Vedení, řízení a koordinace práce sloužící k tomu, aby byl vyvinut produkt, či zajištěna služba. Na př. péče o dodržení požadavků na software.

Správa produktu (Product Management)

Určování, koordinace a péče o vlastnosti produktu v průběhu jeho vývoje. Např. péče o konfiguraci.

Správa zdrojů (Resource Management) Určování, odhady, přidělování a evidence prostředků užívaných při vývoji produktu, či při zajišťování služby. Např. kalkulace.

Potud heslovitý výklad normy. Zajímavé však je uvést další samostatné činnosti, na které se lze z hlediska norem dívat skoro jako na profese a které při tom v naší praxi nejsou běžné, nebo dokonce nejsou příliš známy jako samostatné činnosti. Příklady činností jsou uvedeny dále.

5.2. Činnosti s globální působností

Péče o kvalitu softwaru (software quality assurance, SQA)

Je to naplánovaný úplný systém pravidel, postupů a vzorů pro výběr a provádění všech akcí, jejichž cílem je vytvořit dostatečnou důvěru v to, že softwarový produkt či jeho komponenta splňuje dané požadavky, formulované ve specifikacích pro návrh softwaru. Směrným dokumentem pro péči o kvalitu softwaru v konkrétním softwarovém projektu je **Plán kvality softwaru** (software quality assurance plan, SQAP).

Péče o kvalitu softwaru je jistou analogií kontroly kvality výrobku v průmyslové výrobě a je to pojem zcela běžně užívaný. Podle rozsahu softwarového projektu a charakteru produktu může být péčí o kvalitu pověřena jak skupina, tak i osoba a to buď jako jedinou činností, nebo kumulovaně s jinou činností (vývojovou, dokumentační a j.).

Péče o konfiguraci softwaru

(software configuration management, SCM)

Je to proces, který se zabývá určováním a rozlišováním jednotlivých položek konfigurace v softwarovém systému, řízením jejich distribuce a provádění změn v nich během celého životního cyklu softwarového systému, registrací a poskytováním informací o stavu těchto položek a požadavky na jejich změny a konečně ověřováním úplnosti a správnosti položek konfigurace.

Je to též disciplína technicko-administrativní správy a dozoru k určování a dokumentování funkčních a fyzických charakteristik položek konfigurace, k řízení provádění změn těchto charakteristik a k registraci a poskytování informací o provádění změn a o stavu implementace.

Dokumentem vymezujícím v rámci konkrétního projektu péči o konfiguraci je **Plán péče o konfiguraci softwaru** (Software configuration management plan, SCMP).

Podle rozsahu softwarového projektu a charakteru produktu může být péčí o konfiguraci prověřena jak skupina, tak i osoba a to buď jako jedinou činností, nebo kumulovaně s jinou činností (vývojovou, dokumentační a j.).

5.3. Ověřovací a kontrolní činnosti

Organizační posouzení (Management review)

Je to formální vyhodnocení buď plánu na úrovni celého projektu nebo skutečného stavu projektu ve vztahu k jeho plánovanému stavu. Vyhodnocení provádí jmenovaná posudková komise.

Na jejím zasedání se přezkoumávají plány, normy a směrnice (podle toho, co se má posuzovat), a to buď absolutně (jaká je jejich úroveň, použitelnost a pod.), nebo se porovnává skutečně dosažený pokrok s plánovaným, anebo se přezkoumává obojí. Jako problematická se zaznamená každá oblast, která podle mínění komise taková je.

Odborné posouzení (Technical review)

Je to formální vyhodnocení materiálů softwaru odbornou komisí. Určují se při tom všechny odchylky od specifikací a norem, nebo se dávají doporučení po té, co byla přezkoumána různá alternativní řešení, při čemž v rámci posuzování jednoho materiálu může být provedeno obojí.

Inspekce softwaru (Software inspection)

Účelem je objevit a blíže určit chyby v jakýchkoliv materiálech (včetně programu), které se týkají vyvíjeného softwaru, nikoliv však tvůrčím způsobem nebo stylisticky zasahovat do vlastních prací. Je to pečlivé formální přezkoušení adekvátnosti provedení, které:

(1) Ověřuje, že uvedené materiály splňují své specifikace

(2) Ověřuje, že uvedené materiály vyhovují požadavkům norem, které na ně mají být aplikovány

(3) Určuje odchylky od norem a specifikací

(4) Poskytuje kvantitativní statistická odborně–technická data (např. o chybách a o procesu tvorby softwaru)

Inspekci provádí 3 až 6 inspektorů, jmenovaných ze softwarových odborníků) a jsou vedeny moderátorem, nestranným ke zkoumaným materiálům (ne tedy např. jejich autorem).

Statistická data o nedostatcích se systematicky sbírají a ukládají se v inspekční databázi. Analýza těchto dat má sloužit ke zlepšení jak produktu, tak procesu jeho vývoje a určit efektivnost provádění inspekčních procesů.

Předvedení (Walkthrough)

Účelem je posoudit předváděné materiály softwaru (např. návrh architektury, podrobný návrh, plány a postupy testování či postupy provádění změn). Hlavním účelem je najít najít nedostatky, vynechávky a rozporná místa, vylepšit materiály softwaru a též vzít v úvahu možnost alternativních řešení.

Na předvedení autor podá celkový přehled o materiálu softwaru, který je předmětem předvedení. Po té následuje všeobecná diskuse účastníků, po níž předvádějící „prochází" materiál softwaru podrobně. Během toho se objevují upozornění na chyby, návrhy na změny a zlepšení a vše se zapisuje. Zachycené připomínky se zpracují do zprávy z předvedení, která se poskytne zainteresovaným složkám.

Přínosem předvedení je i vzájemné poskytování pracovních technik, různých stylů práce a znalostí mezi účastníky. Předvedení může ukázat problémy v efektivnosti a čitelnosti programu, problém modularity v návrhu, nebo praktickou netestovatelnost některých podmínek, kladených na návrh.

Funkční kontrola (Functional audit, FA)

Účelem je před dodáním softwaru komisionálně provést nezávislé vyhodnocení softwarových produktů a ověřit při tom, že skutečná funkčnost a výkonové parametry položek jeho konfigurace jsou v souladu se specifikacemi požadavků na software.

Výsledkem musí být:

(1) *Schválení*, nebo

(2) *Podmíněné schválení*, jestliže k nápravě stačí jen konkrétní, jednoduše definované (a dobře patrné) úpravy, po jejichž provedení je možno upustit od opakování kontroly.

(3) *Neschválení*, jestliže materiál softwaru byl shledán jako závažným způsobem nevyhovující.

Fyzická kontrola (Physical audit, PA)

Účelem je před dodáním softwaru komisionálně provést nezávislé vyhodnocení položky konfigurace softwarového produktu, aby se potvrdilo, že složky její dodávané verze

odpovídají její specifikaci. Konkrétně se má ověřit, že software a jeho dokumentace navzájem souhlasí a jsou připraveny k dodání. Výsledkem musí být schválení, podmíněné schválení či neschválení, jako při funkční kontrole.

Průběžná kontrola (In-process audit, IPA)

Provádí se v průběhu etap specifikace a návrhu, dříve než se provede funkční kontrola, aby se ověřilo, že návrh je úplný a bezrozporný. Dále mohou průběžné kontroly být prováděny k ověření souhlasu s aplikovatelnými normami postupů.

5.4. Činnosti přímo související s vývojem

Ladění (Debugging)

Tento proces je chápán v podstatě tak, jak jsme zvyklí. Je to proces, do kterého patří **analýza selhání** softwaru a **oprava chyb**, která ho způsobují. Ladění se zabývá spíše na komponentami softwarového systému, než systémem jako celkem. Jeho cílem je dosáhnout, aby navržený softwarový produkt se choval tak, jak předepisují jeho specifikace. Je obvyklé, že ladění provádí pracovník, který softwarový produkt navrhl.

Testování (Testing)

Je to cílevědomý samostatný proces (oddělený od procesu vývoje softwaru), jehož úkolem je pokusit se způsobit selhání (failure) softwarového produktu s cílem zjistit, že je v něm chyba (fault). Selháním se rozumí skutečnost, že nastala událost, při které systém nebo jeho složka neplní požadovanou funkci v daných mezích, tedy jednoduše řečeno, nejde jen o vznik havarijních stavů, ale o všechny případy, kdy se software nechová přesně tak, jak bylo specifikováno.

Do testování se tedy zahrnuje činnost kontrolní, ale nikoliv opravná. Obvykle netestuje autor programu, ale zvláštní pracovník, nebo skupina, která je na tuto činnost specializována a provádí ji soustavně, nebo v určitém časovém období. Výsledkem testování jsou písemné dokumenty, konkrétně jednotlivé *„zprávy o události při testování"* a *„souhrnná testovací zpráva"*, na základě kterých autoři softwarových objektů odstraňují nedostatky nalezené při testování.

Vlastní proces testování je taxativně rozdělen na tyto tři fáze:

(1) fáze **PLÁNOVÁNÍ TESTOVÁNÍ:**

Patří sem hrubé naplánování přístupu k testování, potřebných zdrojů a rozvrh testování a určení které vlastnosti mají být testovány.

(2) fáze **POŘÍZENÍ SADY TESTŮ:**

Patří sem návrh sady testů a příprava na realizaci navrženého plánu testování

(3) fáze **TESTOVÁNÍ OBJEKTU:**

Patří sem provedení testovacích procedur, zjištění dostatečnosti otestování a vyhodnocení procesu testování a stavu testovaných objektů

Pro potřebu testování se vypracovávají písemné dokumenty, jejichž cílem je mimo jiné i umožnit použití hotových testů v budoucnu při údržbě téhož a vývoji dalších podobných softwarových produktů. Jsou to zejména:

(1) **Plán testování** (test plan) jako dokument nejvyšší úrovně, který uvádí jaké objekty a které jejich vlastnosti budou testovány, jaký přístup a postup bude použit, jaké testovací příklady budou použity a jak budou testy vyhodnocovány, personální zajištění, časový rozvrh, zvážení rizik a další okolnosti.

(2) **Specifikace návrhu testování** (test–design specification), který je v podstatě dílčím, avšak již podrobným popisem testování určitého objektu, neboli rozpracováním plánu testování pro určitý objekt.

(3) **Specifikace testovacích příkladů** (test–case specification), čili popis testovacího příkladu, včetně okolností jak a na jaký objekt může být použit a jaký je správný výsledek provedeného testu.

(4) **Specifikace postupu při testování** (test–procedure specification), která je návodem při provádění testování, včetně počátečního nastavení podmínek a závěrečného zrušení již nepotřebných pomocných souborů, programů a j. a zrušení takových mimořádných (nastavených nebo při testování vzniklých) podmínek a stavů, které jsou trvalého rázu a je třeba je zrušit aktivním zásahem.

6. Závěr

S vědomím toho, že do budoucna se ani Československo nevyhne používání této metodiky, nebo alespoň styku s ní, jsou ve VAKUS Praha vytvářeny interní příručky (zejména [11] až [16]), které v maximální míře respektují výše uváděné normy. Jsou určeny programovým tvůrcům z vlastních řad a mají být přípravou na to, aby tato metodika byla pochopena, postupně se vžila a v přiměřené míře a modifikaci přešla do užívání.

Literatura

[1] ANSI/IEEE Std 729–1983, IEEE Standard Glossary of Software engineering Terminology.

[2] Norma ANSI/IEEE 1002–1987, IEEE Standard Taxonomy for Software Engineering Standards.

[3] ANSI/IEEE Std 730–1984, IEEE Standard for Software Quality Assurance Plans,

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

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

[6] ANSI/IEEE Std 829–1983, IEEE Standard for Software Test Documentation

[7] Norma ANSI/IEEE 1008–1987, IEEE Standard for Software Unit Testing

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

[9] Norma ANSI/IEEE 1012–1986, IEEE Standard for Software Verification and Validation Plans

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

[11] Metodické pokyny pro tvorbu programů v rámci SPT, příručka pro analytiky a programátory, SPT Praha, duben 1992.

[12] Pravidla pro přípravu Plánu kvality softwaru, příručka pro pracovníky z oblasti vývoje softwaru, VAKUS Praha, březen 1992.

[13] Příručka pro sestavování specifikace pro vývoj softwarových produktů, příručka pro uživatele a zadavatele vývoje softwaru, VAKUS Praha, únor 1992.

[14] Doporučená dokumentace pro testování softwaru – popis, příručka pro pracovníky z oblasti vývoje softwaru, VAKUS Praha, duben 1992.

[15] Doporučená dokumentace pro testování softwaru – příklady dokumentů, příručka pro pracovníky z oblasti vývoje softwaru, VAKUS Praha, duben 1992.

[16] Pravidla pro vytváření uživatelské dokumentace softwarových produktů, příručka pro programátory a autory doprovodné dokumentace, VAKUS Praha, únor 1992.

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