MODELOVÁNÍ JAKOSTI SOFTWAROVÝCH PRODUKTŮ V ŘÍDICÍCH SYSTÉMECH

Martin Halva

Fakulta strojního inženýrství VUT v Brně, Technická 2896/2, 616 69 Brno
halva@upei.fme.vutbr.cz

Abstrakt

Příspěvek seznamuje s výsledky disertační práce, která se zabývá modelováním jakosti softwarových produktů v řídicích systémech. Byly zkoumány závislosti interních a externích metrik, které vypovídají o jakosti softwarového produktu. Za použití statistiky vznikly lineární modely závislosti.

Úvod

Jakost počítačových systémů, které hrají ve společnosti stále důležitější úlohu, je v posledních letech předmětem velkého zájmu. Zejména v aplikacích kritických z hlediska bezpečnosti může nesprávné chování počítačových systémů způsobit zvýšené náklady, poškození majetku, případně zdraví lidí nebo dokonce smrt. Jakost softwaru, který tvoří hlavní část těchto systémů, je tedy sama předmětem zvyšujícího se zájmu. Čas a náklady spotřebované na vývoj softwaru informačního systému jsou od počátku 90. let větší než u hardwarových komponentů.

V současné době je jakost vnímána jako schopnost výrobku, systému nebo procesu plnit požadavky zákazníků. Toto pojetí dává volnost v tom, zda se při sledování jakosti zaměříme na softwarový produkt, systém obsahující software nebo na proces vzniku softwaru. Tyto přístupy mají samozřejmě svá specifika. Všeobecně se dnes předpokládá, že jakost produktu je jakostí procesu tvorby do značné míry determinována. Co mají tyto přístupy společné, je jejich zaměření na zákazníka. U softwaru je zákazníkem uživatel. Jakost softwaru je tedy určována relativně ve vztahu k požadavkům (potřebám) jeho uživatele a rozhodně není totožná s jeho technickou dokonalostí. Jediným kritériem je splnění požadavků uživatele.

Průzkum v evropském měřítku (2) ukázal, jak uživatelé a vývojáři softwaru vnímají potřeby a požadavky na hodnocení a certifikaci softwaru. Výsledky průzkumu shrnují následující tři body:

- Certifikace procesu tvorby softwaru sama o sobě není pro garantování jakosti softwarového produktu dostatečná, je nutná také certifikace softwarového produktu.
- Zákazníci jsou ochotni zaplatit až o 20% vyšší cenu za jakostní softwarové produkty.
- Hodnocení softwarového produktu má prvořadou důležitost v rozhodování o vývojové strategii.

Mezi existující postupy hodnocení softwarového produktu patří např. testování metodou černé skříňky (black – box testing), statická analýza (static analysis), kontrola (inspection), měření rozsahu testování (test coverage). Je navrhováno (2) používání uceleného schématu, ve kterém jsou tyto jednotlivé postupy integrovány tak, aby metriky které jsou pro hodnocení jakosti produktu shromažďovány, podávaly ucelenou informaci. Avšak hodnocení

a certifikace jakosti softwarových produktů jsou pouze jedním z mnoha důvodů, proč jsou softwarové metriky shromažďovány. Lze je použít i na odhady nákladů, management nebo pro zhodnocení několika vývojových procesů, nových postupů, programovacích jazyků apod.

1. Řízení

Řízení je obvykle chápáno dvěma poněkud odlišnými způsoby. Za prvé jde o řízení v užším slova smyslu, tj. řízení technologických procesů, které je v anglické literatuře nazýváno „control“. Druhý, širší význam pojmu řízení spočívá v zahrnutí dalších částí výrobního systému včetně rozhodovacího subsystému, kde probíhá řízení včetně uvážení lidských faktorů. Tomuto typu řízení je vyhrazen nejen v angličtině, ale v poslední době už i v češtině, pojem management.

Management zahrnuje koordinované činnosti pro nasměrování a řízení organizace (ISO 9000:2000) - podle (7). Například management jakosti se skládá z koordinovaných činností pro nasměrování a řízení organizace s ohledem na jakost. Naproti tomu řízení jakosti (quality control) je část managementu jakosti zaměřená na splnění požadavků na jakost (ISO 9000:2000) - podle (7). Řízení (control) je tedy částí managementu.

Je vidět, že oba pohledy na řízení spolu souvisejí tak, že management v sobě zahrnuje i řízení v užším smyslu slova. Protože každý ze zmiňovaných přístupů v sobě zahrnuje rozdílné přístupy a postupy, bude se lišit i software pro jejich podporu.

2. Software v řídicích systémech

Z důvodu odlišnosti řídicích systémů pro management a pro řízení procesů v reálném čase bylo nutno definovat oblast zájmu. Disertační práce se ve svém řešení soustředí na software řídicích systémů v reálném čase a ponechává stranou software řídicích systémů pro management. Řídicím systémem je tedy dále myšlen právě řídicí systém na operativní úrovni - řídicí systém ve smyslu anglického „control”.

Tab.1: Přehled programovatelných funkcí v řídicích systémech

Funkční skupina	**Příklady**
Logické řízení - Logické obvody - Časovací jednotky - Čítače	 Člen AND Časovací jednotky zpožďující stav „on“, stav „off“, generující časované impulsy Vzestupné, sestupné, obousměrné
Sekvenční řízení	Sekvenční funkční diagram
Zpracování signálu/dat - Matematické funkce - Manipulace s daty - Zpracování analogových dat	 Základní aritmetické: sčítání, odčítání násobení, dělení Rozšířené aritmetické: třídění, goniometrické funkce Porovnávací: větší, menší, rovno Vyhledávací, uspořádávací, formátovací, přesouvací PID (proporcionálně integrační a derivační regulátory) integrace, filtrování

Funkce rozhraní - Vstup/výstup - Jiné systémy - MMI (rozhraní člověk-stroj) - Tiskárny - Velkokapacitní paměť	analogové, číslicové moduly vstup/výstup Převod BCD (dvojkově kódovaných desítkových čísel) Komunikační protokoly Zobrazovací jednotky, příkazy Zprávy, hlášení Protokolování
Řízení výpočtu (provedení)	Výpočty periodické, řízené událostí
Uspořádání systému	Kontrolování stavu

Tab.1 uvádí příklady funkcí, které se vyskytují v oblasti řídicích systémů. Tyto programovatelné funkce jsou v tabulce shrnuty do funkčních skupin. Z tohoto přehledu (ČSN EN 61131-1) je patrné, jaké skupiny funkcí se dají očekávat v příslušném softwaru. Z uvedených podskupin a příkladů funkcí vyplývá, že v oblasti systémového programového vybavení řídicích systémů je třeba počítat se značným výskytem vstupně-výstupních operací a tedy i vstupních a výstupních dat. Pro modelování jakosti softwaru řídicích systémů bylo nutno zvolit metriky, kterými by bylo možné podchytit důležité vlastnosti sledovaného softwaru. Zvolené metriky musí co nejpřesněji vypovídat o důležitých vlastnostech softwaru tak, aby bylo možné jeho další hodnocení. Bylo tedy nutné, aby byly zvoleny pro experiment takové metriky, které vypovídají o funkčnosti logických a matematických konstrukcí, včetně dalších pomocných funkčností, dále o správnosti vstupních a výstupních dat, funkčnosti uživatelského rozhraní a podobně.

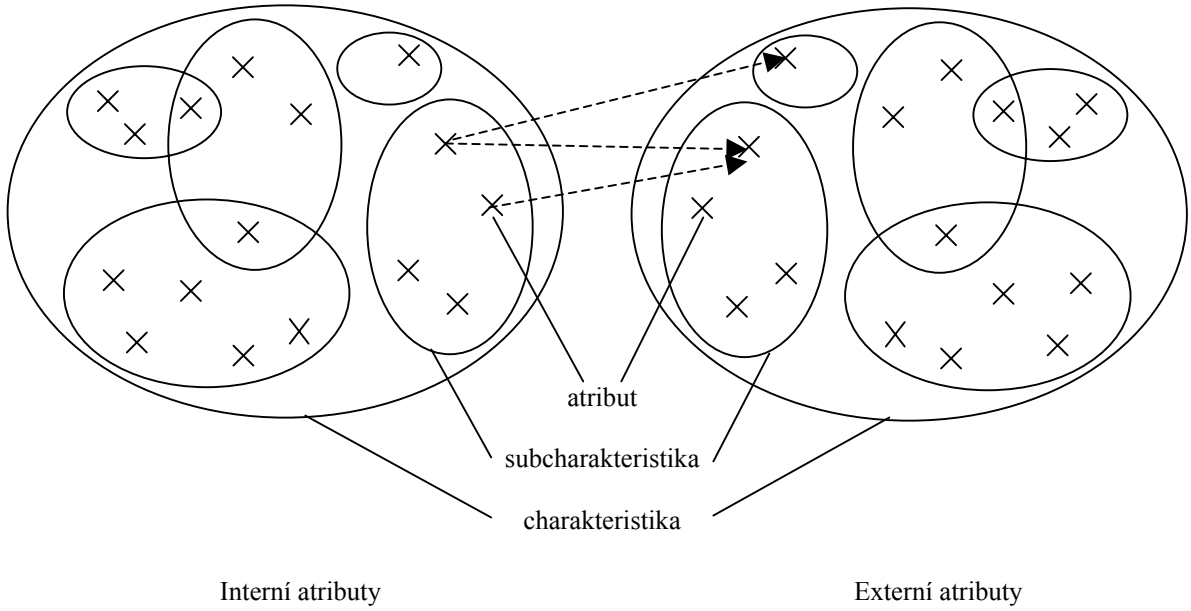
3. Modelování jakosti softwaru

3.1 Atributy a charakteristiky

Charakteristiky jakosti softwaru jsou v (5) definovány takto: „soubor vlastností softwarového produktu, kterým je jeho jakost popisována a hodnocena; charakteristika jakosti softwaru může být zjemňována do mnohonásobného počtu úrovní subcharakteristik."

Úroveň určitého *interního atributu* ovlivňuje hodnotu určitého externího měřítka. Existuje tedy jak externí tak interní aspekt většiny charakteristik. Například bezporuchovost může být měřena externě sledováním počtu chyb v daném časovém intervalu při zkoušení softwaru a interně kontrolou detailní specifikace a zdrojového kódu, čímž hodnotíme úroveň chybnosti.

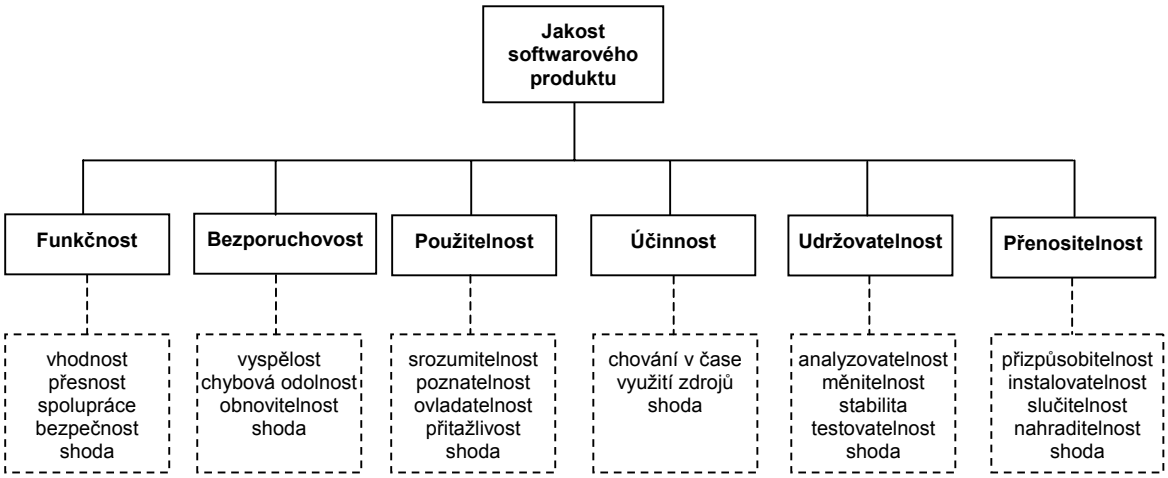
Interní atributy mají tedy být indikátory budoucích externích atributů. Jeden interní atribut může ovlivňovat jednu nebo více charakteristik a také jedna charakteristika může být ovlivňována jedním nebo více atributy. Hierarchická struktura charakteristik a subcharakteristik rozčleňuje atributy jakosti softwaru (viz Obr.1).



Obr.1: Hierarchická struktura charakteristik, subcharakteristik a atributů jakosti softwaru

3.2 Model jakosti softwaru

Model (Obr.2) obsažený v mezinárodní normě ISO 9126 , která je základní normou týkající se zabezpečování jakosti softwaru, byl vytvořen na základě různých modelů jakosti. Výsledkem je množina šesti konzistentních charakteristik, které pokrývají všechny hlavní oblasti. I v případě, že při reálném hodnocení jakosti softwaru je použit jiný model, postupy sběru dat, jejich ukládání a analýzy jsou nezávislé na zvoleném modelu. Každá z uvedených šesti charakteristik (plné obdélníčky v Obr.2) rozdělena na subcharakteristiky (čárkované obdélníčky).



Obr.2: Model jakosti softwarového produktu podle ISO 9126

3.3 Interní metriky

Metrika jakosti softwaru je v (5) definována takto: „kvantitativní stupnice a metoda, která může být použita k určení dosažené hodnoty význačného rysu u konkrétního softwarového produktu.“

Interní metriky měří interní atributy a mohou predikovat externí atributy pomocí analýzy statických vlastností meziproduktu nebo konečného produktu. Při měření interních metrik se používá počet či četnost prvků, ze kterých se software skládá. Tyto prvky se vyskytují například ve zdrojovém kódu (příkazy) nebo v různých typech diagramů (datové toky, přechody stavů apod.). Interní metriky poskytují uživateli, hodnotiteli, ověřovateli a vývojáři možnost hodnotit jakost softwarového produktu a ovlivňovat výslednou jakost ještě před vznikem konečného produktu.

3.4 Externí metriky

Externí metriky softwarových produktů jsou odvozeny z chování systému, kterého je softwarový produkt částí. Zjišťují se testováním, provozováním a sledováním konečného softwaru nebo systému. Externí metriky vycházejí z požadavků na používání softwarového produktu v určitém organizačním a technickém prostředí. Externí metriky poskytují uživateli, hodnotiteli, ověřovateli a vývojáři schopnost hodnotit jakost softwarového produktu během testování a provozu.

3.5 Vztah mezi externími a interními metrikami

Při definování požadavků na jakost softwaru se vytváří seznam podstatných charakteristik a subcharakteristik jakosti. Potom se určí příslušné externí metriky a přípustné rozsahy. Tím se kvantifikují kritéria jakosti, která ověří zda software vyhovuje požadavkům uživatele. Dále se definují interní atributy jakosti softwaru, čímž se při vývoji produktu předpokládá dosažení požadovaných hodnot externích charakteristik jakosti a charakteristik jakosti při používání. Pro měření těchto hodnot se zavedou interní metriky a jejich přípustné rozsahy.

Je nutné, aby se používaly interní metriky, které mají co nejsilnější vztah k cílovým externím metrikám. Tak mohou být předpověděny budoucí hodnoty externích metrik. Všeobecně se soudí, že je obtížné navrhnout rigorosní teoretický model, který postihne silný vztah mezi interními a externími metrikami. Prozatím nezbývá, než se pokusit vztah nalézt experimentálně a stanovit empirický model.

4. Experiment

4.1 Plán experimentu

MĚŘENÉ VELIČINY - METRIKY:
Pro splnění cíle - zjištění vzájemné závislosti charakteristik jakosti softwaru - bylo nutné sestavit plán experimentu. Ten přesně stanoví, které veličiny a za jakých podmínek se budou měřit. S využitím poznatků softwarového inženýrství z (2), (3), (6), (9) byly vybrány následující metriky jakosti softwaru: FSZ, FZ, DNF, CVY, UR, BP, BPS, NVS, TP, TT, McCabe’s, Length – viz tab. 2.

Tab.2: Použité metriky jakosti softwaru

ZKRATKA:	NÁZEV:	VZTAH PRO VÝPOČET:	POZNÁMKA:
FSZ	**F**unkční **s**plnění **z**adání	**počet splněných funkcí ze zadání** / **počet požadovaných funkcí v zadání**	relativní počet splněných funkcí
FZ	**F**unkční **z**ralost	**počet opravovaných funkcí** / **počet požadovaných funkcí v zadání**	relativní počet odstraněných chyb
DNF	**D**robná **nef**unkčnost	**počet drobných nefunkčností** / **počet požadovaných funkcí v zadání**	relativní počet chyb v doplňkových a pomocných funkcích
CVY	**Ch**yby **v**ýstupu	**počet chybných výstupních dat** / **počet požadovaných dat v zadání**	relativní počet chybných výstupních dat
UR	Chyby v **u**živ. **r**ozhraní	**počet chyb v uživatelském rozhraní** / **počet požadovaných funkcí v zadání**	relativní počet chyb v uživatelském rozhraní
BP	Čas **b**ezpor.**p**rovozu	**doba bezporuchového provozu**	čas provozu do první chyby
BPS	**S**třední čas **b**ezpor. **p**rovozu	**střední doba mezi poruchami**	průměrný čas provozu od poruchy do poruchy
NVS	**N**eošetřenost **v**stupu	**počet neošetřených vstupů uživatelských dat** / **počet požadovaných vstupů v zadání**	relativní počet umožněných chybných vstupů
TP	Čas(**t**)**p**rogramo vání	**celkový čas věnovaný programování**	-
TT	Čas (**t**) **t**estování	**celkový čas věnovaný testování**	-
McCabe's	Cyklomatické číslo	**počet hran - počet vrcholů+2**	složitost grafu řízení
Length	Délka programu	**délka zdrojového kódu**	alternativa počtu řádků zdrojového kódu

PODMÍNKY PŘI MĚŘENÍ:

Sledované veličiny bylo nutno zjišťovat při přesně vymezených podmínkách. Je známo, že je obecně obtížné stanovit metriky jakosti, které by byly univerzálně aplikovatelné. Je tedy nutné se omezit na specifické aplikační oblasti, úlohy, programovací jazyky apod.
ÚLOHA: Byla definována jedna typová úloha - zadání, tj. zpracovat jednoduchý program pro výpočet povrchu a objemu jehlanu. Vstupní parametry - rozměry jehlanu - jsou zadány uživatelem na dotaz programu formou textového dialogu a prostředí MS-DOS. Uživatel dále vybere, které parametry jsou požadovány pro výstup a po provedení výpočtu program vypíše výsledky na obrazovku.

PROGRAMÁTOR: Aby bylo možné výsledky experimentu statisticky zpracovat, bylo nutné získat co největší množství programů od srovnatelných programátorů a navíc vzniklých ve srovnatelném prostředí. K tomuto účelu byla využita spolupráce se ZŠ Vratislavovo nám.124 v Novém Městě na Moravě, která má několikaletou tradici s technickou orientací žáků a výukou programování různými metodami a v různých programovacích prostředcích. Pro experiment se podařilo využít závěrečných seminárních zkoušek ve specializované třídě. Seminární zkoušky se zúčastnilo 26 žáků, kteří zpracovávali na počítačích zadanou úlohu. Pro provedení experimentu bylo tedy k dispozici 26 programátorů se srovnatelnou teoreticko-praktickou základnou a zkušenostmi v oblasti programování osobních počítačů ve vyšších programovacích jazycích.

HARDWARE + SOFTWARE: Experiment probíhal v učebně výpočetní techniky vybavené 13 žákovskými personálními počítači Autocont PC P II 333 MHz, 3 GB HDD, 32 MB RAM, 15“ monitor; operační systém Windows 98, programovací jazyk Borland Pascal.
PRACOVNÍ PROSTŘEDÍ: Učebna výpočetní techniky, individuální práce na počítači, ergonomické osvětlení, pedagogický dozor. K dispozici byla kompletní nápověda programovacího prostředí Borland Pascal, manuály, knihovna příkladů a přístup k vlastním programům z minulých projektů a zkoušek.

METODIKA PROGRAMOVÁNÍ: Programátoři používali strukturované programování založené na lineárních programových sekvencích. Používali větvení, cykly, procedury. Metodika determinovaná též zvoleným programovacím prostředím Borland Pascalu - možnost použít krokování, breakpointy, sledování hodnot vybraných proměnných atd. Kompilátor a linker je součástí programovacího prostředí Borland Pascalu. Výstupem z procesu programování a testování měl být zdrojový kód a spustitelný soubor.

METODA MĚŘENÍ: Zvolená metoda měření musela respektovat hlavní účel, tj. zaznamenat zvolené veličiny - hodnoty metrik ve správný okamžik. Pro zaznamenávání naměřených hodnot byl vypracován formulář s použitím normy ISO 9126: Information Technology - Software Quality Characteristics and Metrics, do kterého byly zaneseny zvolené indikátory - metriky s popisem, kdy kterou hodnotu zaznamenávat.

Druhá část experimentu vyžadovala změření interních metrik hotových programů tak, aby je bylo možné vyhodnotit ve vztahu k externím charakteristikám a metrikám zjištěným pomocí zmíněných formulářů. Sběr interních metrik byl uskutečněn za pomoci analytického nástroje – programu QUALMS, který umožňuje analýzu zdrojového kódu programu a zjištění měřitelných parametrů – hodnot metrik (tj. interních). Toto měření bylo realizováno až po ukončení první části experimentu – po odevzdání výsledků práce programátorů a vyplnění formulářů pro sběr externích metrik. Analytický software QUALMS se skládá ze základního

modulu a tzv. Front-End modulů. K dispozici byly moduly Front-End pro programovací jazyky C a Pascal. Analýza probíhala tak, že po otevření zdrojového kódu analyzovaného programu v příslušném Front-End modulu systém QUALMS vygeneroval sérii souborů s tzv. grafy řízení (control-flowgraphs). Tyto soubory slouží následně jako vstup pro výpočet příslušných interních metrik.

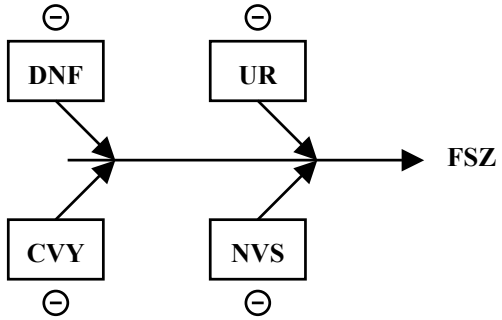
4.2 Vyhodnocení experimentu

ZÁZNAMY:
Z distribuovaných formulářů pro záznam naměřených hodnot externích metrik se vrátilo 23 kusů. To je z celkového počtu 26 programátorů (23/26)% = 88,5%. Druhou část záznamů z experimentu tvoří tabulky naměřených hodnot interních metrik vytvořené pomocí analytického nástroje QUALMS. Z celkového počtu 26 programů jich bylo získáno pro měření interních metrik 16, což je 61,5%. Chybějící programy ve formě zdrojového kódu programátoři nedodali. Všech 16 zdrojových kódů bylo zanalyzováno a výsledkem bylo 16 tabulek interních metrik.

METODA PRO VYHODNOCENÍ:
Cílem je nalezení vztahu mezi interními a externími metrikami. Tento vztah je nutné nejprve analyzovat metodou korelační analýzy. K tomuto účelu byly příslušné interní a externí metriky sestaveny do tabulky a analyzovány statistickým nástrojem SPSS, což je profesionální softwarový nástroj pro statistickou analýzu na počítači. Je například používán k vyhodnocení výsledků průzkumu veřejného mínění nebo statistických závislostí při řízení jakosti v průmyslu.

5. Interpretace výsledků

U proměnných s významným korelačním koeficientem byla na základě Spearmanova testu zamítnuta hypotéza o nezávislosti příslušných proměnných. Přesněji, jednalo se o test nekorelovanosti dvojic veličin na základě Spearmanova testu pořadové korelace. Test je popsán např. v knize Anděl (1). Zamítnutí hypotézy znamená, že dvojice metrik jsou na sobě statisticky významně závislé. S pravděpodobností 95% tedy jedna ovlivňuje druhou. Např. tedy McCabe's/Length ovlivňuje statisticky významně NVS apod. Interpretovatelné závislosti jsou znázorněny formou Ishikawových diagramů „příčina – následek" na obrázcích Obr.3, Obr.4, Obr.5, Obr.6, Obr.7 a Obr.8.

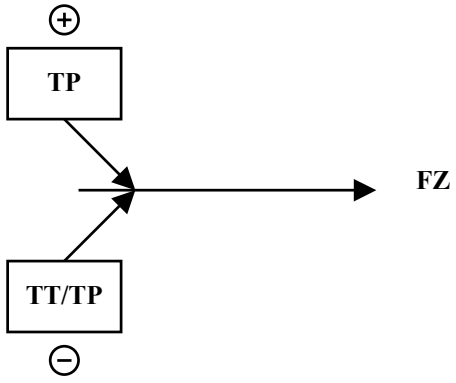


Obr.3: Ishikawův diagram FSZ

Na Obr.3 - FSZ je v Ishikawově diagramu znázorněna závislost Funkčního splnění zadání (FSZ), tj. poměru počtu splněných funkcí k počtu všech funkcí požadovaných v zadání, na čtyřech veličinách – proměnných – DNF, UR, CVY, NVS. Bylo tedy prokázáno, že v daných podmínkách a na dané úloze je funkční splnění zadání ovlivněno drobnými nefunkčnostmi,

chybami v uživatelském rozhraní, chybami na výstupu a neošetřeností vstupů negativně. To znamená, že zvyšující se hodnota jmenovaných charakteristik jakosti způsobuje snížení hodnoty funkčního splnění zadání. Negativní ovlivňování, dané záporným korelačním koeficientem, je znázorněno v obrázku Ishikawova diagramu znaménkem Θ u příslušné charakteristiky.

Ze zjištěné závislosti lze vyvodit závěr, že při vývoji softwaru je pro plné splnění funkcí daných v zadání nutné věnovat se nejen přímo příslušné funkčnosti programu, tj. např. použití správného vzorce pro výpočet objemu a povrchu jehlanu, ale je nutné věnovat patřičnou pozornost správné „prezentaci" výsledků uživateli a správnému „zjišťování" vstupních dat od uživatele, tj. věnovat se dalším ovlivňujícím faktorům. Prakticky to znamená předcházet drobným nefunkčnostem, odstraňovat chyby v uživatelském rozhraní a na výstupu a zabezpečovat správné ošetření vstupů dat do programu.

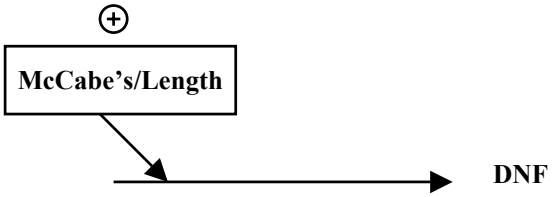


Obr.4: Ishikawův diagram FZ

Na Obr.4 - FZ jsou v Ishikawově diagramu znázorněny dvě příčiny – veličiny TP a TT/TP k následku FZ (Funkční zralost). Funkční zralost jako poměr počtu opravovaných funkcí k celkovému počtu funkcí vyžadovaných v zadání je pozitivně ⊕ ovlivněna délkou programování měřenou časem spotřebovaným k vytvoření „surového" programu a negativně Θ ovlivněna poměrem času testování k času programování.

Z těchto výsledků lze vyvodit závěr, že v daných podmínkách a na dané úloze závisí počet nalezených a odstraněných chyb (Funkční zralost) na čase věnovaném programování a podílu času testování na čase programování. Tedy delší čas věnovaný programování znamená také větší funkční zralost výsledného programu. Dále čím větší je podíl testovacího času na čase programování, tím menší je funkční zralost.

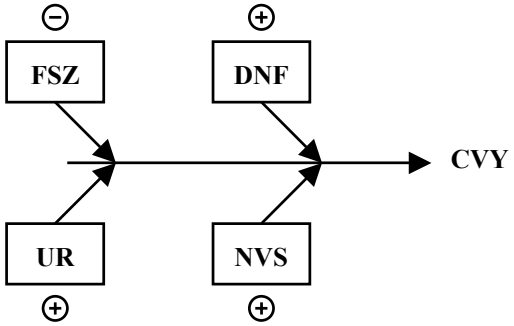
Toto zjištění by si zasloužilo hlubší rozbor. Nabízí se např. vysvětlení, zda větší poměr testování programu není na úkor času programování, tedy zda seriozní systematické programování se svou programovací a testovací fází nepřechází při větším poměru TT/TP při daných experimentálních podmínkách v programování typu „pokus – omyl" (kdy programátor raději testuje než promyšleně programuje). Lze tedy doporučit na základě výsledků experimentu to, aby nebyla omezována ani doba programování, ani testování bez toho, aby byla zvýšena efektivita programování a testování (např. použitím formálního modelu vývoje softwaru nebo počítačové podpory testování – automatizace).



Obr.5: Ishikawův diagram DNF

Obr.5 - DNF znázorňuje závislost DNF – Drobné nefunkčnosti, tj. podílu počtu doplňkových funkcí k počtu funkcí požadovaných v zadání, na charakteristice poměrné složitosti programu měřené metrikou McCabe's/Length. Byla prokázána kladná korelace. Prokázáním vlivu se potvrdil předpoklad, že poměrná složitost McCabe's/Length může být vhodným indikátorem jakosti softwaru.

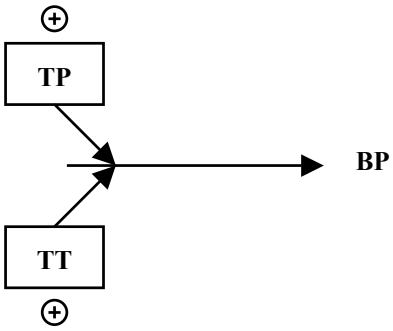
Naproti tomu charakteristika složitosti programu (resp. řídicího flowgrafu) měřená samotnou metrikou McCabe's nevykázala korelaci s žádnou jinou sledovanou charakteristikou (kromě délky programu). Navržený model charakteristiky poměrné složitosti programu (řídicího flowgrafu) se ukázal být relevantním. Prokázal se jeho vliv na drobné nefunkčnosti se vyvíjeném softwaru.



Obr.6: Ishikawův diagram CVY

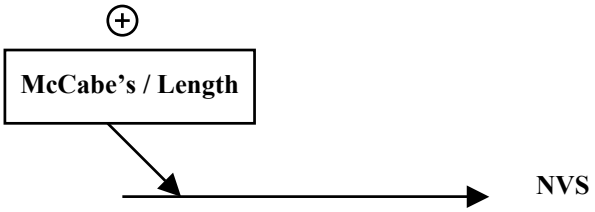
Logicky a vyváženě na první pohled působí závislosti znázorněné na Obr.6 - CVY pro chyby na výstupu. Ukázalo se, že výstupní chyby závisí pozitivně na Neošetřenosti vstupu (NVS), chybách v Uživatelském rozhranní (UR), Drobných nefunkčnostech (DNF) a negativně na Funkčním splnění zadání (FSZ).Výstupní chyby tedy závisí jak na jakosti vstupů, tak na vlastním zpracování dat ve funkcích hlavních i doplňkových.

Nalezená závislost je velmi zajímavá i pro praxi. Lze z ní vyvodit závěr, že je třeba při vývoji softwaru věnovat pozornost všem složkám procesu automatizovaného zpracování dat, tj. vstupu dat, vlastnímu zpracování i prezentaci výsledků v uživatelském rozhraní. Teprve jako celek může být software zřejmě jakostní.



Obr.7: Ishikawův diagram BP

Významným výsledkem korelační analýzy jednotlivých veličin se jeví závislost uvedená na Obr.7 - BP. Je zde vyznačen pozitivní vliv hodnot časů programování (TP) a testování (TT) na charakteristiku bezporuchovosti měřenou Bezporuchovým provozem (BP) – tj. časem do první poruchy. Za daných podmínek a na dané úloze je tedy prokazatelně významný vliv délky programování a délky testování na bezporuchový provoz.
Jako možné vysvětlení se nabízí to, že čím větší úsilí měřené časem věnovaným efektivitě na programování a testování vyvíjeného softwaru, tím delší je bezporuchový provoz, tedy pravděpodobně v hotovém softwaru je méně chyb, resp. více chyb bylo odstraněno.



Obr.8: Ishikawův diagram NVS

Obr.8 - NVS, poslední ze série Ishikawových diagramů, znázorňuje jedinou závislost - závislost Neošetřenosti vstupů (NVS) na charakteristice poměrné složitosti programu (řídícího flowgrafu) měřenou metrikou McCabe's/Length. Jedná se o podobnou situaci jako na Obr.5 - DNF. Zde je tedy další možné využití metriky McCabe's/Length - pro předpověď neošetřenosti vstupů. Vhodnost této nově navržené metriky pro modelování jakosti softwaru se tím zvyšuje.

Vysvětlení vlivu McCabe's/Length na NVS může spočívat v tom, že čím větší je poměrná složitost, tedy jakási hustota složitosti programu vztažená k délce programu, tím větší je pravděpodobnost, že programátor bude mít v komplexním problémovém prostoru sníženou orientaci a opomene ošetřit některé vstupy (vstupní funkce) vůči možným chybám uživatele, resp. chybám vstupních dat.

Na základě zjištěné závislosti (korelace) je možné doporučit používat přesné standardizované postupy programování, ve kterých je ošetření vstupů již zahrnuto jako standardní krok. Ke snížení rizika vzniku chyby programátora lze obecně doporučit použití počítačové podpory strukturovaného programování a formální metody - konceptuální modely vývoje softwaru.

5.1 Empirické modely závislosti

Korelační koeficienty vypočtené pro všechny kombinace jednotlivých naměřených veličin vypovídají jen o tom, zda dvojice proměnných jsou ve vzájemném vztahu, či ne. Pro lepší vyjádření charakteru vzájemné závislosti měřených veličin byly sestrojeny empirické modely závislosti. Naměřenými body byly proloženy regresní přímky a vypočteny jejich rovnice. Ty mohou být považovány za modely jakosti softwaru platné v definovaných podmínkách (viz Tab.3). Bodové odhady regresních koeficientů uvedených v Tab.3 jsou doplněny o intervaly spolehlivosti (konfidenčními intervaly) pro koeficient směrnice.

Pro získání informace o přesnosti hodnot vypočtených z modelu byl každý model doplněn ještě o intervaly spolehlivosti pro vypočtené hodnoty závisle proměnné. Tyto intervalové odhady vytvoří u každého modelu tzv. pás spolehlivosti kolem regresní přímky. Je to pás, jehož šířka pro pevně zvolenou hodnotu x udává s jistou spolehlivostí (bylo voleno $\alpha = 0,05$) nepřesnost vypočtené hodnoty závisle proměnné z modelu. Pásy spolehlivosti přesahují svým rozsahem možnosti tohoto článku, proto zde nejsou uvedeny.

Tab.3: Rovnice regresních přímek – modely

Popis modelu	**Regresní rovnice**	**Interval spolehlivosti pro směrnici**
Závislost funkčního splnění zadání na drobných nefunkčnostech, chybách výstupu, chybách v uživatelském rozhraní a na neošetřenosti vstupů.	FSZ = -1,537 DNF + 1,070	<-1,881; -1,193>
	FSZ = -1,052 CVY + 1,058	<-1,482; -0,621>
	FSZ = -0,534 UR + 0,825	<-0,961; -0,106>
	FSZ = -0,404 NVS + 0,826	<-0,696; -0,111>
Závislost funkční zralosti na poměrném čase testování a na čase programování.	FZ = -1,332 TT/TP + 2,310	<-5,153; 2,490>
	FZ = $1,824 \cdot 10^{-5}$ TP + 1,830	<$-1,78 \cdot 10^{-5}$; $5,43 \cdot 10^{-5}$>
Závislost drobných nefunkčností a neošetřenosti vstupů na metrice poměrné složitosti programu.	DNF = 0,499 McCabe's/Length + $9,34 \cdot 10^{-2}$	<0,027; 0,971>
	NVS = 1,026 McCabe's/Length – $7,93 \cdot 10^{-2}$	<0,111; 1,941>
Závislost chyb výstupu na funkčním splnění zadání, chybách v uživatelském rozhraní, drobných nefunkčnostech a neošetřenosti vstupů.	CVY = -0,524 FSZ + 0,696	<-0,739; -0,309>
	CVY = 0,593 UR + 0,206	<0,375; 0,812>
	CVY = 0,853 DNF + 0,124	<0,464; 1,243>
	CVY = 0,345 NVS + 0,230	<0,158; 0,532>
Závislost doby bezporuchového provozu na čase programování a čase testování.	BP = $5,951 \cdot 10^{-2}$ TP + 506,933	<0,020; 0,099>
	BP = 0,208 TT + 1113,588	<0,087; 0,328>

Závěr

Při definování požadavků na jakost softwaru se vytváří seznam podstatných charakteristik a subcharakteristik jakosti. Potom se určí příslušné externí metriky a přípustné rozsahy. Tím se kvantifikují kritéria jakosti, která vypovídají o tom, zda software vyhovuje požadavkům uživatele. Dále se definují interní atributy jakosti softwaru, které lze měřit již v ranných stádiích jeho vzniku, např. při analýze, návrhu nebo kódování. Pro měření těchto hodnot se zavedou interní metriky a jejich přípustné rozsahy. Je nutné, aby se používaly interní metriky, které mají co nejsilnější vztah k cílovým externím metrikám. Tak mohou být do jisté míry předpověděny budoucí hodnoty externích metrik dříve, než bude možno testovat a měřit hotový produkt.

V popisované disertační práci byla navržena a experimentálně na modelovém příkladu ověřena metodika pro vyhodnocování vztahu mezi externími a interními metrikami jakosti softwaru. Rovnice regresních přímek vzniklé při experimentu představují empirický model jakosti zkoumaného softwaru. Ze zjištěných závislostí lze vyčíst mnoho poznatků i o procesu vývoje softwaru. Některé postřehy jsou popsány už v komentářích k Ishikawovým diagramům, které znázorňují vzájemné korelace zjištěných metrik.

Při vyhodnocování výsledků experimentu byly využity statistické metody korelační analýzy, lineární regrese a analýzy odlehlých bodů. Prezentované modely jsou pochopitelně platné jen pro experimentem definované podmínky. Proto je při aplikaci metodiky na konkrétní software určitého řídicího systému nutné vždy vycházet z příslušných reálných podmínek (programovací jazyk, typ úlohy, typ operačního systému, zkušenost programátorů apod.). Modely se budou ve svých koeficientech lišit, protože budou odrážet specifika daného konkrétního případu.

Literatura:

1. Anděl, J. Matematická statistika. Praha: SNTL a Alfa, 1978
2. Bache, R., Bazzana, G. Software Metrics for Product Assessment. McGraw-Hill, 1994. ISBN 0-07-707923-X
3. Card, D. N., GLASS, R.L. Measuring Software Design Quality. New Jersey: Prentice-Hall, 1990
4. ČSN EN 61131-1. Praha: Český normalizační institut
5. ČSN ISO/IEC 9126: Informační technika, Hodnocení softwarového produktu – charakteristiky jakosti a návod pro jejich používání. Praha: Český normalizační institut, 1993
6. Garmus, D., Herron, D. Measuring the Software Process - a practical guide to functional measurements. Prentice Hall, 1996
7. Komentované vydání návrhů norem ISO/DIS 9000:2000. Praha: Český normalizační institut, 2000
8. Lacko, B. Trendy informačních systémů po roce 2000. In Jakost a informační systémy. Brno: VUT v Brně, 1999. ISBN 80-214-1321-2
9. Smith, D. J., Wood, K.B. Engineering quality software. Essex, England: Elsevier Science Publishers, 1989. ISBN 1-85166-358-4
10. Vaníček, J. Měření a hodnocení jakosti informačních systémů. Praha: Česká zemědělská univerzita v Praze, 2000. ISBN 80-213-0667-X