STATISTICKÉ MODELOVÁNÍ JAKOSTI SOFTWARU

Martin Halva

Ústav metrologie a zkušebnictví, Fakulta strojního inženýrství, Vysoké učení technické v Brně, Technická 2896/2, 616 69 Brno, halva@umz.fme.vutbr.cz

Úvod

Mezinárodní norma ISO 9000:2000 vymezuje *jakost* jako stupeň splnění *požadavků* souborem tzv. *znaků jakosti*, které jsou obsaženy ve *výrobku*, *procesu* nebo *systému*. Tato obecná definice vlastně klade důraz na schopnost výrobku, procesu nebo systému plnit požadavky svých zákazníků. Toto pojetí dává volnost v tom, zda se při sledování jakosti zaměříme na softwarový produkt nebo na proces vzniku a vývoje softwaru. Všeobecně se dnes předpokládá, že jakost produktu je jakostí procesu jeho tvorby významně determinována. Avšak, z důvodu velké variability procesu tvorby softwaru (tvůrčí proces individuálních osobností – analytiků, návrhářů, programátorů atd.), bez nezávislého hodnocení jakosti konečného softwarového produktu není možné dát jasnou odpověď na otázku, zda je dosažená jakost dostatečná.

Z tohoto důvodu není při hodnocení jakosti softwaru použití samotných norem ISO 9000 dostatečné. Ty poskytují pouze důležitý rámec. Je nutné se zabývat jakostí samotného softwarového produktu. Protože se jedná o nehmotný produkt – výsledek duševní činnosti člověka – jsou měřitelné vlastnosti jakosti softwaru na první pohled obtížně představitelné. Situace se poněkud vyjasní, pokud je budeme slovně charakterizovat, tj. přiřazovat jim tzv. *charakteristiky jakosti*, které navíc rozdělíme na interní a externí. Nejdříve přistoupíme k samotné definici charakteristik jako základu pro modelování jakosti softwaru.

1. Modelování jakosti softwaru

1.1 Metriky jakosti softwaru

Jakost softwaru se popisuje pomocí charakteristik a subcharakteristik (11, 8). V softwarovém produktu je obsažena v jeho atributech. Čím ji ale měřit? Ne vždy lze měřit přímo atribut nebo charakteristiku. Navíc musí být jasné, v jakých jednotkách a jakým způsobem měříme. K tomuto účelu byl zaveden a rychle se také vžil pojem *metrika*. Dnes se v praxi používá nejen v oblasti softwaru, ale v řízení jakosti vůbec. Avšak předtím, než si metriku definujeme, bude vhodné zmínit se o terminologickém problému, který je s používáním pojmu metrika v oblasti řízení jakosti spojen.

Terminologická poznámka:
„Existuje nejednotnost v definicích a výkladech pojmů *metrika* a *míra*. Obě slova se v teorii měření používají v jiném smyslu, než je tomu v jiných oblastech matematiky. Příslušnou definici z teorie měření lze nalézt v (11). Pojem *metrika* je používán v souladu s mezinárodní normalizací (2). V teorii metrických prostorů se používá slovo *metrika* jako zobecnění pojmu vzdálenosti dvou bodů. Je to funkce, která přiřazuje dvojici bodů nezáporné reálné číslo (nulu pouze tehdy, jsou-li oba body totožné), je symetrická a splňuje trojúhelníkovou nerovnost.

Slovo *míra* se používá v tzv. teorii míry k zobecnění pojmu délky úsečky, obsahu obdélníka a objemu kvádru z těchto jednoduchých množin na obecnější třídu tzv. měřitelných množin.“

1.2 Interní metriky

Metrika jakosti softwaru je praktickým způsobem popsána v (2) takto: „kvantitativní stupnice a metoda, která může být použita k určení dosažené hodnoty význačného rysu u konkrétního softwarového produktu.“ Znamená to, že metriky se používají pro reálné zjištění měřených hodnot atributů jakosti určitým prakticky proveditelným způsobem.

Tak jako jsou atributy rozčleňovány na interní a externí, je k nim pochopitelně nutné rozlišovat také interní a externí metriky. Interní metriky se s výhodou používají během fází návrhu a programování softwarového produktu, který je zatím například v podobě specifikace nebo zdrojového kódu. Interní metriky tak měří vnitřní vlastnosti produktu. Jejich účelem je predikovat externí jakost a jakost při používání.

1.3 Externí metriky

Míry externích metrik softwarového produktu závisejí na chování systému, kterého je 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 reálnou schopnost hodnotit jakost softwarového produktu během testování a provozu.

1.4 Vztah mezi externími a interními metrikami

Je jasné, že se při definování požadavků na jakost softwaru musí používat interní metriky, které mají co nejsilnější vztah k cílovým externím metrikám. Jen tak mohou být předpovězeny budoucí hodnoty externích metrik. Všeobecně se soudí, že je obtížné navrhnout rigorózní teoretický model, který postihne silný vztah mezi interními a externími metrikami. *Není však vyloučeno sestavení empirického modelu.* A právě zde se otevírá pole působnosti pro *použití statistiky*.

Při zabezpečování jakosti softwaru obecně je nedostatečná možnost, jak jakost měřit. Existují sice modely, jak popsat jakost softwaru pomocí charakteristik – jeden z modelů se dokonce stal součástí mezinárodní normy ISO 9126 a souvisejících. Charakteristiky jako funkčnost, bezporuchovost, použitelnost, účinnost, udržovatelnost, přenositelnost, které jsou dále strukturovány do subcharakteristik, postihují z různých hledisek jakost uvažovaného softwaru. Problémy však mohou nastat, pokud se pokusíme přiřadit charakteristikám nebo subcharakteristikám jakosti softwaru příslušné metriky. Existuje mnoho publikací zabývajících se metrikami, některé metriky byly zahrnuty i do norem. Problémem však zůstává jejich nízká vypovídající schopnost nebo obtížná použitelnost v praxi. K odstranění těchto problémů by jistě pomohlo, kdyby použití metrik jakosti softwaru bylo experimentálně ověřeno. To se neobejde bez *statistického vyhodnocení.*

2. Experiment s objektově orientovanými programy

V rámci disertační práce autora tohoto článku byl proveden experiment, jehož cílem bylo zjistit na praktickém příkladu, zda existuje závislost mezi externími a interními metrikami

jakosti softwaru včetně indikátorů složitosti. K vyhodnocování tohoto vztahu byla navržena a použita metodika založená na postupné aplikaci vybraných statistických metod. Metodika má pracovní název MJSW (modelování jakosti softwaru).

Pokud by hledaná závislost mezi externími a interními metrikami existovala, bylo by možné do určité míry usuzovat na jakost softwaru už z měření v ranných stádiích jeho tvorby, tj. např. ve fázi kódování nebo dokonce návrhu. V těchto ranných fázích vzniku softwaru je odhalení chyb, které by později při provozu mohly způsobit poruchy, velmi žádoucí nejen z důvodu prevence havárií, ale i proto, že odstraňování chyb v těchto prvotních stádiích je *několikanásobně* snadnější a lacinější, než později.

2.1 Cíl experimentu

Cílem tohoto experimentu bylo ověřit metodiku modelování jakosti softwaru (MJSW) na rozsáhlejších programech. Proto byla navázána spolupráce s pracovníky firmy e-Fractal, s.r.o. Tato firma se zabývá vývojem softwaru pro on-line vyhledávání a rezervaci leteckých spojení. Rozsah a nároky na provoz takového softwarového systému jsou značné. Bližší popis je možné nalézt níže v popisu podmínek při měření. Poznatky získané na základě analýzy vypovídají jak o možnostech použití metodiky, tak částečně o jakosti vyvíjeného softwaru - pomocí objektivně měřitelných znaků jakosti. Byly měřeny programové moduly tohoto objektově orientovaného systému pro vyhledávání a rezervaci leteckých spojení – moduly nazvané ADROT, AMAGAL, AMADEUS, EXTREM, IAM, XMLGAL.

2.2 Podmínky při měření

Hardware a software:

- LAN (NT, WIN2000): PC Pentium III - 500 MHz – 1 GHz, 256 MB RAM,
- spojení do rezervačního systému: X.25, TCP/IP, XML,
- vývojové prostředí: CINCOM Visual Works 5i.3 (jazyk Smalltalk),
- objektově orientovaný databázový systém: GEMSTONE/S 5.1.5.

Úloha:

- vývoj, programování a testování modulů objektově orientovaného systému pro vyhledávání a rezervaci leteckých spojení,
- cca 2000 objektových tříd,
- cca 300 MB databáze,
- cca 176 tis. záznamů v databázi,
- cca 22,5 mil. letů uloženo v databázi,
- cca 2000 dotazů (vyhledání) v provozu denně.

Programátoři:

- projektové týmy o velikosti 2-4 programátorů,
- vzdělání: VŠ (ČVUT, ČZU PEF),
- dovednosti: Smalltalk (4 roky praxe), TCP/IP, obecné znalosti XML, přenosových protokolů, OO paradigmatu atd.

Metodika programování:

- týmová práce na projektech,
- ranní tzv. „Stand-up meeting“ (diskuse o plánu dne),
- na konci pracovní doby tzv. „Go-home meeting“ (zhodnocení, priority, účast šéfa),

- „extrémní programování“ (důraz na testování),
- automatické noční spouštění sestavování aplikace a testů.

Pracovní prostředí:
- 3 programátorská pracoviště po 2 programátorech u jednoho PC – vše v 1 místnosti,
- komunikace neustálá – spontánního typu,
- stoly umístěny v rozích místností pro určitý pocit soukromí,
- flexibilní přepážky podle potřeby,
- žaluzie,
- osvětlení.

Metoda měření:
- automatický sběr interních metrik pomocí SW nástroje OOM (Aoki, Japonsko),
- sběr externích metrik do připraveného formuláře (ISO 9126) pomocí interview,
- modelování jakosti softwaru (Halva – disertační práce).

2.3 Interní metriky

Z výše uvedené stručné charakteristiky objektově orientovaného programu lze určit měřitelné vlastnosti zdrojového kódu objektově orientovaného programu jako je počet tříd, počet metod ve třídě, počet atributů ve třídě, počet proměnných ve třídě atd. Soubor těchto tzv. základních objektově orientovaných metrik je uveden v tab. A-1.

Tab. A-1 – Základní interní metriky

Zkratka	Význam	Popis – definice
LOC	*lines of code*	*LOC* (*C*) vyjadřuje počet řádků zdrojového kódu v dané třídě *C*.
NOM	*number of methods*	*NOM* (*C*) vyjadřuje počet metod v dané třídě *C*.
NIM	*number of instance methods*	*NIM* (*C*) vyjadřuje počet instančních metod v dané třídě *C*.
NCM	*number of class methods*	*NCM* (*C*) vyjadřuje počet metod třídy v dané třídě *C*.
NOA	*number of attributes*	*NOA* (*C*) vyjadřuje počet atributů v dané třídě *C*.
NIV	*number of instance variables*	*NIV* (*C*) vyjadřuje počet instančních proměnných v dané třídě *C*.
NCV	*number of class variables*	*NCV* (*C*) vyjadřuje počet proměnných třídy v dané třídě *C*.

Chidamber a Kemerer navrhli další metriky (viz tab. A-2) pro složitost objektově orientovaného programu na základě postupu vývoje objektově orientovaných systémů. Metriky navržené Chidamberem a Kemererem se týkají především složitosti návrhu tříd objektů. V podstatě podchycují tzv. intermodulární složitost (viz 11). Proto se hodí pro měření rozsáhlých programů použitých při experimentu.

Tab. A-2 – Chidamberovy-Kemererovy interní metriky

Zkratka	Význam	Popis – definice
WMC	*weighted methods per class*	Váha metod pro třídu je pro každou třídu *C* definována jako součet složitostí c_j všech *n* metod třídy *C* takto: $WMC(C) = \sum_{j=1}^{n} c_j$ Jednotlivé složitosti c_j se stanovují jako složitosti imperativních programů.
DIT	*depth of inheritance tree*	Hloubka stromu dědičnosti *DIT* (*C*) je pro každou třídu *C* definována jako maximální délka (počet hran) cesty od kořene k vrcholu ve stromu dědičnosti odpovídajícímu v hierarchii tříd dané třídě.
NOC	*number of children*	Představuje počet přímých potomků *NOC* (*C*) vrcholu představujícího ve stromu dědičnosti danou třídu *C*.
CBO	*coupling between object classes*	Spřažení objektů mezi třídami *CBO* (*C*) udává pro každou třídu *C* počet tříd, na historii jejichž objektů třída závisí tím, že volá jejich metody nebo používá jejich instanční proměnné.
RFC	*response for a class*	Odezva třídy *RFC* (*C*) je definována jako počet metod, které jsou buď metodami třídy *C* nebo patří do jiné třídy, ale některá metoda třídy *C* je volá. Jde tedy o počet metod, jejichž provedení může být potencionálně vyvoláno tím, že některý objekt třídy *C* přijal zprávu.
LCOM	*lack of cohesion of methods*	Jde o nedostatečnou soudržnost tříd v metodách vyjádřenou pomocí počtu dvojic spřažených a nespřažených metod dané třídy. Dvě metody M_1 a M_2 téže třídy *C* pokládáme za spřažené, pokud průnik $I_1 \cap I_2$ jejich instančních proměnných je neprázdný, tj. mají společnou alespoň jednu instanční proměnnou. Značí-li *p* počet spřažených dvojic metod třídy *C* a *q* počet nespřažených dvojic metod třídy *C*, metrika *LCOM* je definována takto: pro $p \geq q$: $LCOM(C) = 0$, pro $p < q$: $LCOM(C) = q - p$. Za dostatečnou soudržnost je tedy považována situace, kdy počet spřažených dvojic je alespoň tak velký, jako nespřažených. Další zvyšování soudržnosti již není považováno za přínos.

Sběr interních metrik (základních i Chidamberových-Kemererových) byl prováděn pomocí softwarového nástroje OOM, který provádí automatickou analýzu softwaru vytvořeného v programovacím jazyce Smalltalk a generuje soubory s hodnotami metrik. Autorem tohoto nástroje je japonský odborník Aoki Atsushi (http://www.sra.co.jp/people/aoki/).

2.4 Externí metriky

Externí metriky na rozdíl od interních neměří strukturální vlastnosti softwaru, ale snaží se podchytit přímo jeho chování při provozu. Sběr externích metrik je proto prováděn při testování softwaru. Podle metodiky modelování jakosti softwaru, která je popsána v disertační práci autora, byly použity následující externí a časové metriky (viz tab. A-3).

Zvláštní význam mají metriky časové (čas spotřebovaný na programování a čas spotřebovaný na testování) – daly by se považovat za metriky procesu vzniku softwaru. Při sběru externích metrik bylo snadné zjistit i hodnoty těchto časových metrik, proto jsou zahrnuty jako doplňkové mezi externí metriky. Dále byla zjišťována jejich schopnost predikovat chování softwaru při provozu tak, jako u interních metrik.

Tab. A-3 – Externí a časové metriky

ZKRATKA	NÁZEV	VZTAH PRO VÝPOČET	POZNÁMKA
FSZ	**F**unkční **s**plnění **z**adání	**počet splněných funkcí ze zadání** / **počet funkcí požadovaných v zadání**	relativní počet splněných funkcí
FZ	**F**unkční **z**ralost	**počet opravovaných funkcí** / **počet funkcí požadovaných v zadání**	relativní počet odstraněných poruch
DNF	**D**robná **nef**unkčnost	**počet drobných nefunkčností** / **počet funkcí požadovaných v zadání**	relativní počet poruch v doplňkových a pomocných funkcích
CVY	**Ch**ybovost **v**ýstupu	**počet chybných výst. datových prvků** / **počet požadovaných výst. datových prvků**	relativní počet chybných výstupních datových prvků
UR	Chyby v **uživ. r**ozhraní	**počet poruch v uživatelském rozhraní** / **počet funkcí požadovaných v zadání**	relativní počet poruch v uživatelském rozhraní
BP	**B**ezporuchový **p**rovoz	**čas bezporuchového provozu**	čas provozu do první poruchy
BPS	**S**třední čas **b**ezporuch. **p**rovozu	**doba pozorování** / **počet poruch**	průměrný čas provozu od poruchy do poruchy
NVS	**N**eošetřenost **v**stupu	**počet chybně zadaných datových prvků** / **počet pokusů o chybné zadání dat. prvku**	relativní počet umožněných chybných vstupů dat
TP	Čas (**t**) **p**rogramování	–	celkový čas spotřebovaný na programování
TT	Čas (**t**) **t**estování	–	celkový čas spotřebovaný na testování

2.5 Způsob výpočtu interních metrik na základě teorie

Interní objektově orientované metriky byly měřeny automaticky a detailně zaznamenávány do poměrně rozsáhlých souborů, které obsahovaly jednotlivé metriky pro každou objektovou třídu modulu. Pro každý měřený softwarový modul byl vygenerován jeden soubor metrik. Šlo o měření softwarového systému pro rezervaci leteckých spojení na internetu. Pro další statistickou analýzu naměřených dat bylo potřeba z dílčích metrik každé objektové třídy vypočítat souhrnné metriky celého měřeného modulu.

Souhrnné základní interní metriky pro celý modul jsou tedy (*m* je počet tříd v modulu):

$$LOC = \sum_{i=1}^{m} LOC(C_i),\ NOM = \sum_{i=1}^{m} NOM(C_i),\ NIM = \sum_{i=1}^{m} NIM(C_i),\ NCM = \sum_{i=1}^{m} NCM(C_i),$$

$$NOA = \sum_{i=1}^{m} NOA(C_i),\ NIV = \sum_{i=1}^{m} NIV(C_i),\ NCV = \sum_{i=1}^{m} NCV(C_i).$$

Souhrnné metriky bylo třeba vypočíst i z Chidamberových-Kemererových interních metrik, jejichž hodnoty byly změřeny také pro každou třídu jednotlivých modulů. Ne u všech Chidamberových-Kemererových interních metrik lze však použít výpočet součtu. Metriky *DIT* a *LCOM* jsou pouze ordinálního typu a navíc ani neměří počty srovnatelných prvků. Výpočet součtu by tedy znamenal porušení pravidel teorie měření. U metrik *DIT* a *LCOM* jsem použil stanovení souhrnné hodnoty pomocí maxima z jednotlivých hodnot naměřených pro všechny třídy příslušného modulu.

Souhrnné Chidamberovy-Kemererovy interní metriky pro celý modul jsou tedy (*m* je počet tříd v modulu):

$$WMC = \sum_{i=1}^{m} WMC(C_i),\ DIT = \max(DIT(C_i)),\ NOC = \sum_{i=1}^{m} NOC(C_i),\ CBO = \sum_{i=1}^{m} CBO(C_i),$$

$$RFC = \sum_{i=1}^{m} RFC(C_i),\ LCOM = \max(LCOM(C_i)).$$

Výše uvedenými způsoby byly vypočteny všechny potřebné souhrnné interní metriky. Každý modul je charakterizován jednou sadou souhrnných interních metrik. Jejich hodnoty byly dále použity při statistickém vyhodnocení korelační analýzou. Nalezené závislosti metrik odhalily zajímavou skutečnost. Na první pohled upoutalo velké množství závislostí interních metrik na jiných interních metrikách. Týká se to jak základních, tak i Chidamberových-Kemererových interních metrik. Příslušné korelační koeficienty lze nalézt v korelační matici v tab. A-6 (I. část) v oddělených přílohách disertační práce.

Z toho jsem učinil závěr, že souhrnné základní interní metriky definované na základě úvahy pomocí součtů a souhrnné Chidamberovy-Kemererovy interní metriky definované na základě teorie (literatury) pomocí součtů a maxim, mají malou schopnost popsat strukturální vlastnosti měřeného softwaru z většího počtu různých pohledů.

2.6 Úprava výpočtu interních metrik

Lze říci, že různé souhrnné základní metriky vypočtené pomocí součtů a také různé souhrnné Chidamberovy-Kemererovy metriky vypočtené pomocí teorií doporučovaných součtů dávají v různých podobách stále stejnou informaci o vlastnostech softwaru, jenom jinak vyjádřenou. Jiná situace je u metrik *DIT* a *LCOM*, u kterých teorie doporučuje místo součtů používat maxima. U nich vzájemná závislost s jinými interními metrikami nebyla prokázána, byla však s 95% spolehlivostí prokázána jejich vzájemná závislost jedné na druhé. Relativně lépe než ostatní se z tohoto pohledu chová také interní metrika *NCV*, která je závislá „jen“ na 4 jiných interních metrikách.

Z uvedených důvodů jsem se rozhodl najít vhodnější způsob výpočtu souhrnných interních metrik. Použil jsem dosud neuvažovanou alternativu – **medián**. Medián je statistická charakteristika, kterou lze použít i na data, která nemají symetrické rozdělení. Jeho další výhodou je, že medián je méně citlivý na případné odlehlé hodnoty, které by mohly zkreslovat hodnoty souhrnných metrik. Navíc medián lze použít i pro metriky ordinálního typu.

Souhrnné základní a Chidamberovy-Kemererovy interní metriky pro celý modul jsou (vlnovka označuje medián, $i = 1 \ldots m$, m je počet tříd v modulu):

$$LOC = \tilde{L}OC(C_i),\ NOM = \tilde{N}OM(C_i),\ NIM = \tilde{N}IM(C_i),\ NCM = \tilde{N}CM(C_i),$$

$$NOA = \tilde{N}OA(C_i),\ NIV = \tilde{N}IV(C_i),\ NCV = \tilde{N}CV(C_i).$$

$$WMC = \tilde{W}MC(C_i),\ DIT = \tilde{D}IT(C_i),\ NOC = \tilde{N}OC(C_i),\ CBO = \tilde{C}BO(C_i),$$

$$RFC = \tilde{R}FC(C_i),\ LCOM = \tilde{L}COM(C_i).$$

Z nové korelační matice (tab. A-8 v oddělených přílohách disertační práce) byl vidět značný úbytek nežádoucích vzájemných korelací interních metrik. V původní korelační matici bylo běžně možné k jednotlivým interním metrikám nalézt 7 až 9 (pro metriku *LOC* dokonce 10) statisticky významných korelačních koeficientů pro závislosti na jiných interních metrikách. V nové korelační matici, vypočtené pro mediány, se u interních metrik vyskytují pouze 2 až 4 korelace s jinými interními metrikami (u metriky *LOC* se počet korelací snížil na 6).
Z tohoto faktu vyvozuji, že použití mediánů je daleko vhodnější, než použití součtů. Také u metrik *LCOM* a *DIT* je z tohoto pohledu výhodnější použití mediánů, protože narozdíl od použití maxim nejsou na sobě tyto dvě metriky vzájemně závislé (není statisticky významná korelace). Další vyhodnocení – analýzu výsledků – jsem proto provedl již jen s použitím mediánů pro výpočet všech souhrnných interních metrik.

3. Výsledky experimentu

Z výsledků korelační analýzy vyplynulo, že neošetřenost vstupu (*NVS*) je nepřímo úměrná počtu proměnných v objektové třídě (*NIV*), počtu atributů v objektové třídě (*NOA*) a počtu metod ve třídě (*NOM*). Bezporuchový provoz (*BP*) je nepřímo úměrný délce času testování (*TT*). Jeho střední doba (*BPS*) je rovněž nepřímo úměrná délce času testování (*TT*) a navíc ještě hloubce stromu dědičnosti (*DIT*). A konečně chybovost výstupu (*CVY*) je nepřímo úměrná odezvě objektové třídy (*RFC*) a počtu instancí metod ve třídě (*NIM*).

Při vytváření regresních přímek bylo nutné přiznat, že regresní analýza není v případě modelů se závisle proměnnou *NVS* bez problémů proveditelná. Naměřené hodnoty se pohybovaly jen v úrovních 1 a 0. Navíc z šesti provedených měření nebyla hodnota ve dvou případech k dispozici. To by samo o sobě hovořilo spíše pro použití jiné metody analýzy, než regrese – například kontingenční tabulky. Na druhou stranu – uvážím-li, že proměnná *NVS* (neošetřenost vstupu) je stanovována jako podíl počtu chybně zadaných datových prvků ku počtu pokusů o jeho chybné zadání, je jasné, že naměřené hodnoty 0 a 1 jsou pouze hraničními hodnotami intervalu všech hodnot, které může metrika *NVS* nabývat. Je také jasné, že hodnoty uvnitř tohoto intervalu jsou racionální čísla. Z tohoto důvodu se domnívám, že regresní model je možné sestrojit, ale nepovažuji jej za konečný výsledný model.

Analýza reziduí prokázala existenci statisticky významně odlehlých bodů, které bylo možné za účelem zpřesnění modelu vyloučit, u modelů *CVY-NIM* a *CVY-RFC*. U obou modelů jde vždy o 3. bod. Po vypuštění odlehlých bodů jsem musel regresní modely znovu přepočítat. Nové modely, které se samozřejmě oproti původním změnily, jsou uvedeny v grafech *Regrese 1a*, *Regrese 2a*. Opakovaná analýza reziduí pro nové regresní modely žádné další odlehlé body neprokázala.

Při pohledu na výsledné grafy lineárních závislostí lze říci, že výsledky vcelku nejsou v rozporu s obecnými zákonitostmi tvorby softwaru pomocí objektově orientované

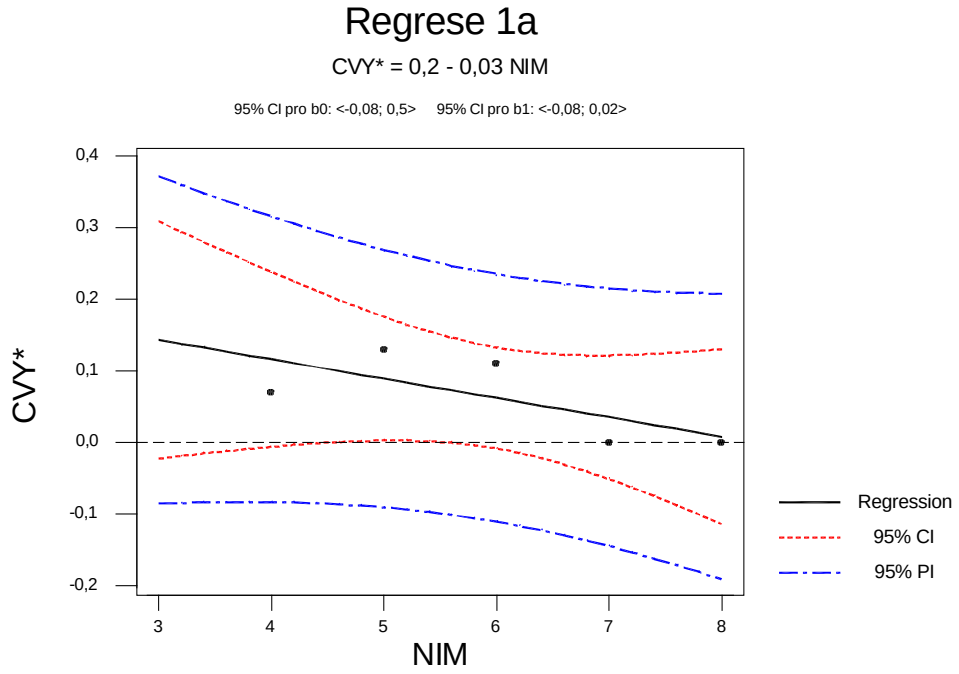
technologie a vypovídají o jakosti analyzovaného softwaru. Nyní rozeberu zjištěné skutečnosti pro každou skupinu závislostí (pro každou závisle proměnnou) podrobněji.

3.1 Modely neošetřenosti vstupů (NVS)

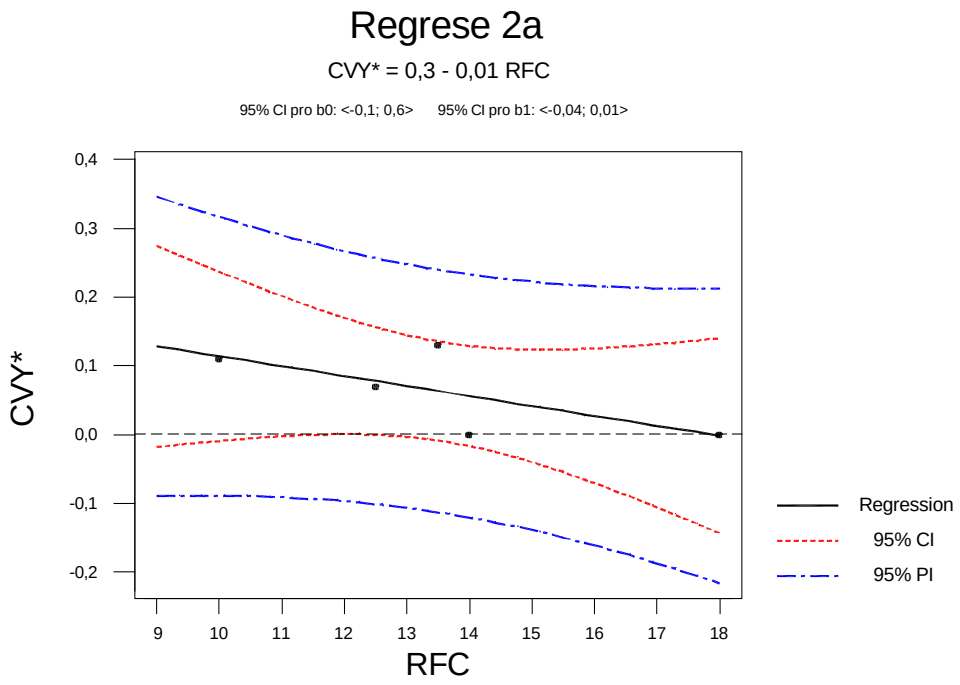
Ponechám-li pro tento okamžik stranou pochybnosti, které byly o tvorbě regresních modelů se závisle proměnnou *NVS* vysloveny výše, lze (grafy *Regrese 6* až *Regrese 8* v oddělených přílohách) formulovat následující zajímavé tvrzení. Neošetřenost vstupů je tím menší, čím větší je v objektových třídách počet konstrukčních prvků, konkrétně atributů (metrika *NOA*), metod (metrika *NOM*) a instančních proměnných (metrika *NIV*). Je tedy možné říci, že tyto tři modely zachycují v kladném smyslu trend, kdy v objektových třídách roste počet konstrukčních prvků, z nichž některé pravděpodobně zabezpečují správné ošetření vstupů vůči nesprávným či nepřípustným hodnotám. Tím, že roste počet všech zmíněných konstrukčních prvků, lze předpokládat, že se zvyšuje i počet konstrukčních prvků potřebných pro správné ošetření vstupů. Není zde sice splněna podmínka dostačující, ale je splněna alespoň podmínka nutná.

3.2 Modely chybovosti výstupů (CVY)

Pro modely se závisle proměnnou *CVY* upravené analýzou reziduí *(*grafy *Regrese 1a*, *Regrese 2a)* lze formulovat následující tvrzení. Čím větší počet instančních metod (indikuje metrika *NIM*) je v objektových třídách obsaženo anebo čím víc vlastních či jiných metod na objektové třídy reaguje (metrika *RFC*), tím menší chybovost je na výstupu. Naopak je tedy možné říci, že model v kladném smyslu zachycuje následující trend: zvyšující se počet všech zmíněných typů výpočetních prvků či odkazů v softwaru zvyšuje i počet implementovaných výpočetních prvků či odkazů, jejichž absence by měla podstatný vliv na zhoršení výstupů z programu. Opět je pro toto tvrzení splněna alespoň podmínka nutná.



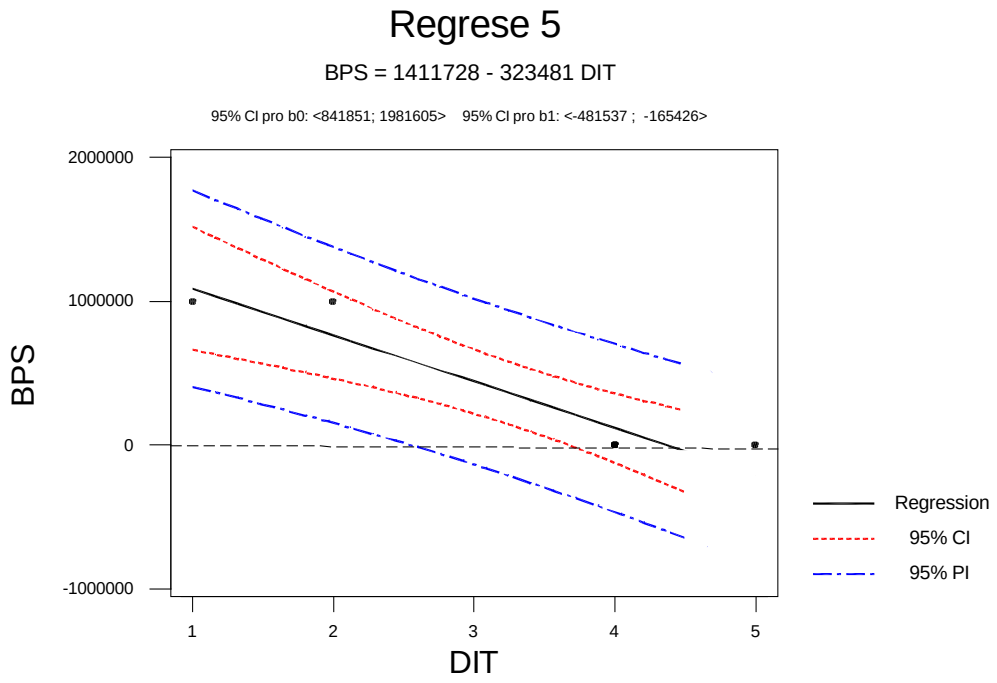
Obr. 1: Regrese 1a



Obr. 2: Regrese 2a

3.3 Modely bezporuchového provozu (BPS, BP)

Pro model se závisle proměnnou *BPS* a nezávisle proměnnou *DIT* (graf *Regrese 5*) lze formulovat následující tvrzení. Střední doba bezporuchového provozu je tím delší, čím menší je hloubka vnoření stromu dědičnosti (tj. menší strukturální složitost softwaru vyjádřená metrikou *DIT*). To odpovídá zkušenostem při programování, kdy programátor je schopen udržet v paměti a zaměřit svoji pozornost vždy jen na konečný rozsah programové struktury (v tomto případě stavba a funkce objektů, metod, proměnných – vlastních a zděděných). Se zvětšujícím se rozsahem softwaru se zvyšuje hodnota hloubky stromu dědičnosti a roste riziko chyb, které programátor může (ale nemusí) udělat.



Obr. 3: Regrese 5

Rozpaky vzbuzuje jen poslední skupina nalezených závislostí – závislost bezporuchového provozu (metrika *BP*), respektive jeho střední hodnoty (metrika *BPS*) – na době testování (metrika *TT*). Odpovídající grafické znázornění je v oddělených přílohách disertační práce uvedeno v grafech *Regrese 3* a *Regrese 4*. Očekával bych spíše přímou úměrnost než nepřímou. Se zvyšující se délkou doby testování by logicky měla růst i doba bezporuchového provozu, respektive jeho střední hodnota.

Příčiny, že tomu tak není, mohou být různé a zasloužily by si prověření v dalších experimentech. Například může jít o klesající část složitější závislosti, kdy bezporuchovost nejdříve skutečně stoupá s prodlužující se délkou testování, ale po dosažení určitého maxima již delší doba testování nepřispívá ke zvýšení bezporuchovosti, ale ze zatím neznámých příčin začne naopak klesat.

Dalším možným vysvětlením může být, že naměřená data padla náhodou do takových bodů, že došlo k opakovanému zachycení extrémů na obou stranách a nalezená závislost, protože je sestavena jen ze šesti bodů, neodpovídá přesně skutečnosti. Data o bezporuchovém provozu mohla být také zjištěna s nedostatečnou spolehlivostí. Metoda řízeného rozhovoru se sice snaží zajistit co největší míru objektivity zjišťovaných dat, ale nemůže dosáhnout spolehlivosti, jakou by mělo získání dat přímo ze záznamů jakosti, které nebyly při měření k dispozici. K vyvrácení či potvrzení pozorovaného jevu a právě formulovaných hypotéz může dojít jen opakovaným měřením nebo lépe – naměřením většího počtu hodnot z širšího intervalu.

Závěr

Statistické modelování jakosti softwaru je možné, vyžaduje však přesné vymezení podmínek platnosti modelu. Navržená metodika modelování se skládá z následujících po sobě jdoucích kroků:

1. Korelační analýza založená na testování hypotézy o nezávislosti zkoumaných metrik pomocí Spearmanova korelačního koeficientu.
2. Sestavení modelů závislosti externích a interních metrik, u kterých byla nalezena statisticky významná korelace, pomocí regresní analýzy.
3. Zpřesnění modelů analýzou odlehlých bodů pomocí t-testu.
4. Vyjádření přesnosti výsledného modelu pomocí intervalů spolehlivosti pro regresní koeficienty a pásů spolehlivosti pro hodnoty vypočtené z modelu.

Literatura:

1. ALIFORNI, S.S. - MATARAZZI, C. *A Flexible Metric for Software Auditing Activities.* In Information Systems Auditing. Elsevier Science Publishers, 1983. ISBN 0-444-86778-3
2. *ČSN ISO/IEC 9126*
3. DEUTSCH, M.S. - WILLIS, R.R. *Software Quality Engineering.* Englewood Cliffs, NJ: Prentice-Hall, 1988
4. HALSTEAD, M.H. *Elements of Software Science.* Amsterdam: North–Holland, 1977
5. HALVA, M. Jakost softwaru užívaného ve firmách. In sborník z celostátní konference *Tvorba softwaru 2000*. Ostrava: Tanger, 2000. ISBN 80-85988-49-6
6. HALVA, M. Modelování jakosti softwarových produktů v řídicích systémech. In sborník z celostátní konference *Tvorba softwaru 2002*. Ostrava: Tanger, 2002. ISBN 80-85988-74-7
7. HALVA, M. *Modelování jakosti softwaru pro informační systémy.* Teze disertační práce. Brno: VUT v Brně, 1998, 38 s.
8. HALVA, M. Quality modelling of software products. In sborník z mezinárodní konference *JAKOST – QUALITY 2002*. Ostrava: Dům techniky a VŠB-TU, 2002. ISBN 80-02-01494-4
9. *ISO/IEC 9126-1*
10. MC CABE, T.J. A Complexity Measure. *IEEE Transactions on Software Engineering*. December 1976, 2, no. 4
11. VANÍČEK, J. *Měření a hodnocení jakosti informačních systémů*. Praha: ČZU PEF ve vydavatelství CREDIT, 2000, 212 s. ISBN 80-213-0667-X