Modelování a predikce spolehlivosti programového vybavení

Jan Chochola

1 Úvod do problematiky

Spolehlivost je jednou z nejdůležitějších složek kvality software. Kromě kvality produktu zajímá obě strany, dodavatele software i jeho (budoucího) uživatele, také termín a cena. Čas a cenu lze snadno měřit a vyjádřit, kvalitu ne. Proto je kvalita opomíjeným (nebo odsouvaným) aspektem.

Dosavadní přístup je orientován spíše na dodavatele a je založen na pokusech spočítat chyby, defekty a/nebo opravy programu. Takových měřítek lze dobře použít pro srovnání s jinými projekty (např. počet chyb/1 000 řádků). Správnější však je zjišťovat zbývající chyby, které se navenek projevují nějakou úrovní spolehlivosti. Ta se již nevztahuje ke statickému návrhu, ale k provozu systému a je tedy spíše dynamickým jevem.

U technického vybavení se projevuje efekt fyzického stárnutí, u software je zdrojem chyb návrh nebo jeho implementace a jejich odstraněním se jich jednorázově zbavíme. Při tom mohou vzniknout chyby nové. Poruchy u programového vybavení také minimálně souvisí s vlastní „výrobou"; kopie programů jsou stále stejné.

1.1 Základní pojmy

Porucha (failure) znamená odchylku výsledku činnosti programu od zadání. Proto je to vlastnost dynamická.

Chyba (fault) je nedostatek programu, který při jeho provádění za určitých podmínek vede k poruše. Je tedy vlastností programu a ne jeho chování. Vzniká chybou programátora.

Spolehlivost se obvykle analyzuje v závislosti na čase. *Doba výpočtu* nebo též *prováděcí čas* (execution time) programu je doba, kterou spotřebuje procesor pro vykonání instrukcí tohoto programu. *Kalendářní čas* (calendar time) je obdobou běžného času. *Hodinový čas* (clock time) představuje interval od spuštění po ukončení činnosti programu na počítači. Doba výpočtu pro analýzu spolehlivosti vyhovuje nejlépe; od ní je však v praxi nutno odvodit zpět reálné hodnoty času kalendářního, případně hodinového. Hodinový ani prováděcí čas nezahrnuje dobu vypnutí systému. (Například za 1 kalendářní týden je 50 hodin hodinového času a 25 hodin prováděcího času.)

Vstupní proměnné existují vně programu a jsou programem využity. Dohromady tvoří *vstupní prostor*, výběr jejich hodnot *vstupní stav*, který má určitou pravděpodobnost výskytu. Obdobně jsou zavedeny *výstupní proměnné*, *výstupní stav* a *výstupní prostor*. Prostředí, ve kterém je systém využíván, je popsáno provozním profilem. Ten je definován množinou typů chodů (funkcí, které program může provádět) a jejich pravděpodobnostmi.

Nejčastěji je *spolehlivost software* definována jako pravděpodobnost bezporuchového provozu programu v daném prostředí po daný čas. Dalším možným vyjádřením je *intenzita poruch*, která se snáze vyhodnocuje.

1.2 Modelování

Spolehlivost je ovlivněna těmito základními faktory: vznikem chyb, odstraněním chyb a prostředím. Vznik chyb závisí na charakteristice kódu (nový/modifikovaný, rozsah, použité nástroje, zkušenost návrháře). Odstranění chyb je závislé na čase, provozním profilu, kvalitě oprav. Konečně prostředí je určeno přímo provozním profilem.

Veličiny jsou pravděpodobnostní, což vede k popisu pomocí náhodných procesů. Mají většinou charakter nehomogenního Poissonova procesu. Poissonův stochastický proces je náhodný proces, který sleduje počet nějakých událostí od počátku sledování do určitého času (a splňuje některé další podmínky). Je charakterizován rychlostí λ. Parametry modelu je možno určit:

- odhadem ze statistických údajů o poruchách
- predikcí z vlastností produktu a procesu vývoje (není třeba program spouštět)

2 Vybrané modely

Základní model založený na době výpočtu (Basic Execution Time Model, BETM) se vyznačuje zejména těmito vlastnostmi:

- obecně přijatelně přesně předpovídá
- je jednoduchý a snadno pochopitelný
- nejvíce propracovaný a nejčastěji používaný v praxi
- jeho parametry mají jasný fyzikální význam a lze je vztáhnout k informacím o programu a vývojovém prostředí ještě před spuštěním
- vyhovuje i pro vyvíjené systémy (systémy, které výrazně mění svůj rozsah) úpravou poruchových časů na hodnoty, jaké odpovídají přítomnosti celého kódu

Logaritmický Poissonův model založený na době výpočtu (Logarithmic Poisson Execution Time Model, LPETM) má vysokou platnost předpovědí již na počátku testovací fáze. Doposud není vyvinuta metoda svázání parametrů s charakteristikami programu před spuštěním. Je o něco pesimističtější než BETM jak v hodnotě intenzity poruch, tak v nákladech (čas, síly, peníze) pro dosažení cíle.

Modely mají společný efektivní duální přístup k poruchovému chování. Modelování probíhá v prováděcím čase a ten je převeden společným aparátem do času kalendářového.

2.1 Složka doby výpočtu

Pro BETM je počet chyb při testování konečný, pro LPETM (a pro nekonečné testování) nekonečný. BETM předpokládá lineární pokles počtu chyb až k nule, LPETM exponenciálně klesající.

Pro intenzitu poruch v závislosti na počtu zjištěných poruch u BETM platí vztah

$$\lambda(\mu) = \lambda_0 \left(1 - \frac{\mu}{\nu_0} \right),$$

kde λ_0 je počáteční intenzita, ν_0 celkový počet poruch pro nekonečný čas a μ je průměrný nebo očekávaný počet zjištěných poruch v daném čase. Pro intenzitu poruch u LPETM:

$$\lambda(\mu) = \lambda_0 \exp(- \theta \mu),$$

kde θ se nazývá logaritmický dekrement intenzity poruch (rychlost poklesu intenzity s každou objevenou a opravenou poruchou).

Pro model BETM se předpokládá spíše rovnoměrný pracovní profil; velmi nerovnoměrný vede k nižšímu sklonu křivky poklesu intenzity. Model LPETM vyhovuje lépe programům se silně nerovnoměrným rozložením profilu: chyby v častěji používaných funkcích jsou dříve detekovány, takže křivka rychleji poklesne, ale později intenzita klesá pomaleji (až velmi pomalu).

Pro časovou doménu (dobu výpočtu τ) a model BETM platí:

$$\mu(\tau) = \nu_0 \left[1 - \exp(- \frac{\lambda_0}{\nu_0} \tau) \right] \quad \text{a} \quad \lambda(\tau) = \lambda_0 \exp(- \frac{\lambda_0}{\nu_0} \tau).$$

Obdobně pro LPETM:

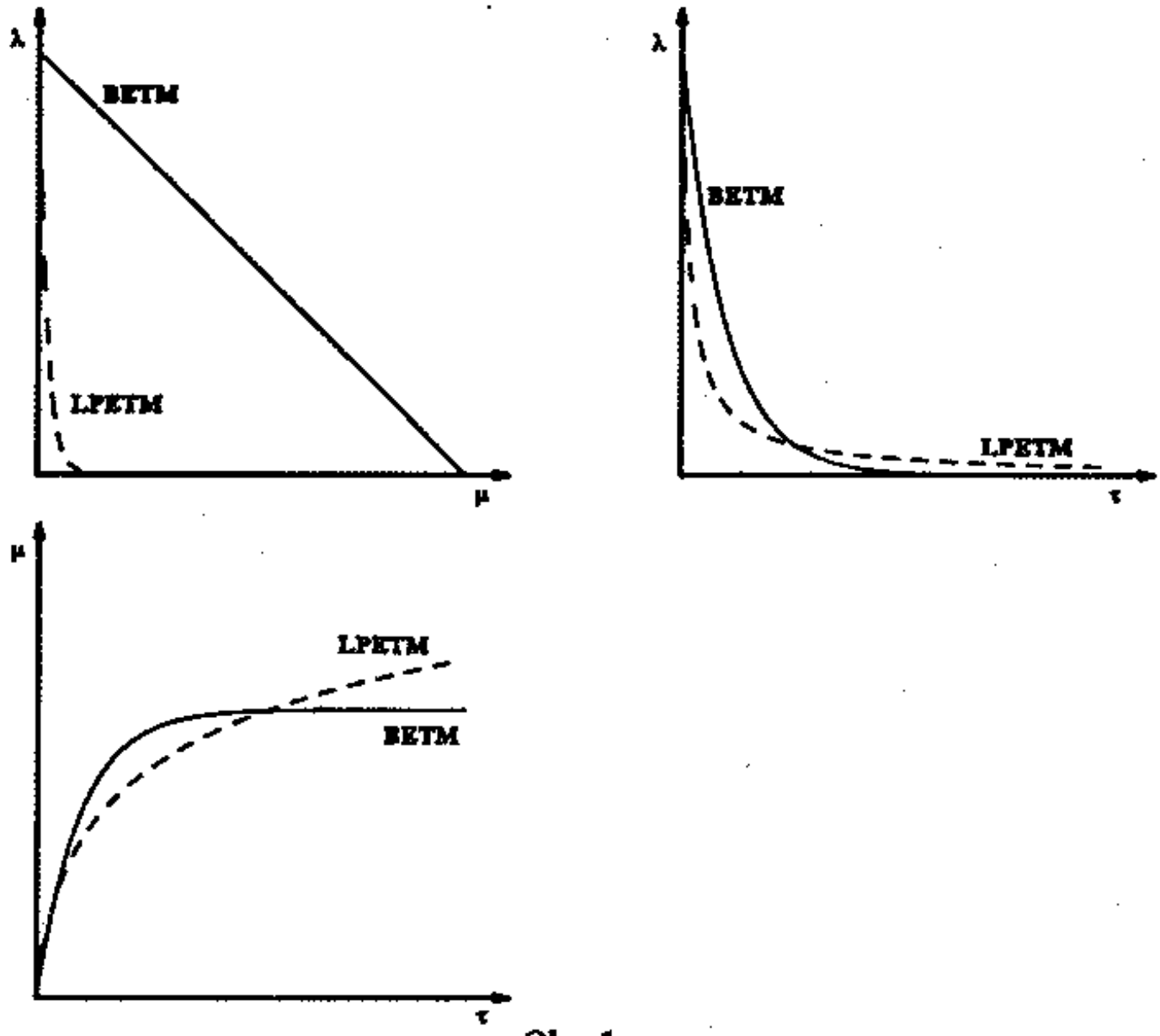
$$\mu(\tau) = \frac{1}{\theta} \ln(\lambda_0 \theta \tau + 1) \quad \text{a} \quad \lambda(\tau) = \frac{\lambda_0}{\lambda_0 \theta \tau + 1}.$$

Charakteristiky a vzájemné vztahy ilustrují grafy na obr. 1.

Označíme-li λ_F konečnou (požadovanou) intenzitu poruch, λ_P současnou intenzitu, $\Delta\mu$ očekávaný počet zjištěných poruch pro dosažení požadované spolehlivosti a $\Delta\tau$ dobu k tomu potřebnou, pak pro odvozené hodnoty platí:

BETM: $$\Delta\mu = \frac{\nu_0}{\lambda_0} (\lambda_P - \lambda_F) \quad \text{a} \quad \Delta\tau = \frac{\nu_0}{\lambda_0} \ln \frac{\lambda_P}{\lambda_F},$$

LPETM: $$\Delta\mu = \frac{1}{\theta} \ln \frac{\lambda_P}{\lambda_F} \quad \text{a} \quad \Delta\tau = \frac{1}{\theta} (\frac{1}{\lambda_F} - \frac{1}{\lambda_P}).$$



Obr. 1

2.2 Složky kalendářového času

Během testů a oprav je kalendářový čas nejdůležitější veličinou (kdy to bude hotovo, ...). Je ho možno získat z prováděcího času pomocí okamžitého poměru, který je předurčen omezenujícími zdroji pro testy a odstraňování chyb. Zdroje jsou určeny většinou v ranných fázích projektu a jejich pozdější doplnění většinou není efektivní.

Testovací fáze probíhá často zhruba následujícím způsobem: Zpočátku je objevováno velké množství poruch a testování je třeba přerušit, protože úzkým místem je rychlost oprav (*F*ailure correction). Později se limitujícím stane kapacita testovacího týmu (failure *I*dentification). Nakonec se úzkým profilem stane kapacita výpočetního systému (*C*omputer). Kalendářová složka je založena na tomto modelu procesu ladění a zahrnuje (index r představuje jednotlivé zdroje I, F a C):

– zdroje *použité* při provozu systému po danou dobu a při ošetření vzniklých poruch (0_r použití/hodina CPU, μ_r použití/porucha)

– *dostupné* zdroje (plánovaný počet P_r)

– stupeň *využití* jednotlivých zdrojů ρ_r

Zatížení zdroje χ_r je lineárně proporcionální k době výpočtu a k počtu nalezených poruch podle vztahu

$$\chi_r = 0_r \tau + \mu_r \mu .$$

Pro opravu závisí zatížení jen na počtu nalezených poruch. Protože počet poruch je funkcí času, je zatížení zdroje ve skutečnosti funkcí času. Mezikrok „nalezených poruch" dává lepší náhled na fyzikální podstatu a zároveň umožňuje sbírat data nezávisle na modelu.

Za předpokladu konstantního množství a využití zdrojů je okamžitý poměr kalendářového času t a prováděcího času τ dán zatížením a kapacitou limitujícího zdroje:

$$\frac{dt}{d\tau} = \frac{1}{P_r \rho_r} \frac{d\chi_r}{d\tau} = \frac{0_r + \mu_r \lambda}{P_r \rho_r} .$$

Tento poměr je třeba počítat pro každý omezující faktor zvlášť. Poměr konverguje k asymptotě $0_r / P_r \rho_r$. Křivka složená z maxim těchto tří průběhů může mít některé z tří přechodů: *FI, IC, FC*. Jí také odpovídá průběh poměru obou časů. Přechody jsou hladké.

3 Stanovení parametrů

3.1 Složky doby výpočtu

Predikcí lze stanovit jen parametry λ_0 a ν_0 pro BETM.

Celkový počet poruch lze odhadnout před testem z počtu *vlastních chyb* programu ω_0 dělením *redukčním faktorem B*. Počet vlastních chyb je nejvíce ovlivněn velikostí programu. Díky tomu lze odhad počtu vlastních chyb získat jako násobek velikosti programu (měřené například ve zdrojových instrukcích) a hustoty vlastních chyb (podle statistik se pohybuje hustota vlastních chyb od zhruba 100 chyb/K řádků po prvních úspěšných překladech až k 1,5 chyby/K řádků v provozní fázi). Lze využít i jiných měřítek velikosti a/nebo složitosti programů (např. Halsteadovu míru).

Redukční faktor B představuje poměr odstranění čistých chyb k počtu zjištěných chyb v nekonečném čase. *Celkový počet* chyb je většinou větší než čistý počet díky zavlékání nových chyb při opravách. Během testů se velikost poměru B pohybuje mírně pod hodnotou 1 (podle [Mu87] zhruba 0,92 až 0,99; průměr 0,955), zatímco v provozní fázi je nestálá (ale pro testy již není důležitá). Při odhadu hodnoty poměru je třeba uvažovat:

– stupeň poznání vztahů mezi poruchami způsobenými společnou chybou

- detekovatelnost chyb vzhledem k dostupným informacím o poruchách
- rozsah na testech nezávislé inspekce kódu
- rozlišení vztahů mezi jednotlivými chybami
- rozsah zavlékání nových chyb při opravách

Počáteční intenzitu poruch λ_0 lze odhadnout podle vztahů

$$\lambda_0 = f K \omega_0, \qquad f = \frac{r}{I},$$

kde veličina f je tzv. *lineární rychlost provádění programu*, r průměrná rychlost zpracování instrukcí a I počet instrukcí (strojových) programu. Lineární rychlost představuje rychlost provádění programu, kdyby byly jeho instrukce seřazeny lineárně. Rychlost chyb $f\,\omega_0$ je průměrná rychlost projevení se chyb při lineárním zpracování. Veličina K je *poměr odhalení chyb* (fault exposure ratio), který představuje pravděpodobnost, že vykonání instrukce vede k poruše (neboli vztah mezi intenzitou chyb a rychlostí, se kterou se chyby projeví). Zahrnuje následující fakta:

- program není lineární, ale má mnoho smyček a větví
- vzhledem ke změnám stavu programu se chyba může, ale nemusí projevit (například vyhodnocení chybné podmínky if i > 0 se od if i > 1 liší jen pro i=1)

V současné době je nutno K určit z charakteristiky obdobného míry programu nebo se smířit s chybou průměrné hodnoty (která není extrémně velká). Podle tabulky 5.6 v [Mu87] se hodnoty pohybují zhruba od 10^{-7} do 10^{-6} se středem cca $4{,}2.10^{-7}$ poruch/chybu.

3.2 Složky kalendářového času

Parametry jsou společné pro oba modely. Většinou je limitujícím faktorem osoba provádějící opravy. Je-li osoba účastna jak identifikace, tak opravy poruch, je třeba ji započítat dvakrát. Vytváření hlášení o poruchách souvisí obvykle s identifikací a je třeba je zahrnout do použití zdroje. Příprava a plánování testů by měla proběhnout *před* započetím testů a do zatížení zdroje se nezapočítává. Pokud je však součástí testů, je třeba je zahrnout do μ_I a 0_I (identifikační práce/porucha a práce/CPU čas).

Plánované množství zdrojů musí stanovit správce projektu. Osoby se jen počítají, pracovní dobu a její délku je třeba vzít v úvahu jen při výpočtu dostupného strojového času a převodu kalendářových hodin na dny nebo data.

Nejlepší odhad počtu opravujících osob je počet programátorů na plný úvazek při programování. Opravy jsou předávány jednotlivým osobám podle jejich „pole působnosti". Pro identifikaci jsou naopak všechny osoby ekvivalentní (každý může testovat a analyzovat cokoliv).

Při využití zdrojů je důležité jeho maximum v období, kdy je zdroj limitujícím faktorem. Pro identifikaci je ρ_I rovno jedné. Je-li kapacita počítače věnována jen testování,

nevzniká mnoho jiných požadavků a jeho využití se také blíží jedné. Využití pro osoby, které chyby opravují, je odvozeno od maximální délky fronty požadavků u každého z nich. Je-li P_{m_Q} je pravděpodobnost, že fronta oprav pro žádnou osobu nepřekročí délku m_Q, pak platí

$$\rho_F = (1 - P_{m_Q}^{1/\lambda_F})^{1/m_Q} .$$

Parametry použití zdrojů charakterizují použití zdrojů jako funkce prováděcího času a objevených poruch. Jsou to:

0_C: průměrná délka strojového času spotřebovaná na jednotku doby výpočtu,

0_I: průměrná doba identifikace spotřebovaná na jednotku prováděcího času,

μ_C: průměrný strojový čas na poruchu,

μ_F: průměrná doba opravy na poruchu,

μ_I: průměrná doba identifikace na poruchu.

Jedná se o míry spotřeby zdrojů a jsou závislé zejména na úrovni programátorů, složitosti úkolů a na prostředí pro vývoj a ladění. Přesně lze hodnoty určit jen na daném projektu. Pro podobné projekty a podobné podmínky lze aplikovat i hodnoty z jiných projektů.

Ukazuje se, že parametry použití zdrojů jsou stejné pro široké třídy projektů. Čisté hodnoty uvádí tabulky 5.11 až 5.13 [Mu87]; pro ilustraci uvedu jejich střední hodnoty: μ_F 4,5 člověkohodin, 0_I 5,91 čh, μ_I 1,22 čh, 0_C 2,43 CPU hodin a μ_C 1,08 CPUh.

4 Aplikační postupy

4.1 Trvání testů systému

Kalendářový čas t se skládá nejvýše ze tří období s různým limitujícím zdrojem. Délku každé z těchto period je třeba vypočíst zvlášť a pak je sečíst:

$$t = \sum_r \frac{\Delta\chi_{rr}}{P_r \rho_r} .$$

Veličina je $\Delta\chi_{rr}$ spotřeba zdroje r po dobu, kdy je omezujícím faktorem. Není obecně totožná s celkovou spotřebou zdroje pro dosažení cílové spolehlivosti $\Delta\chi_r$. P_r je celkové dostupné množství a ρ_r jeho využití. Pro potenciální přechody mezi intervaly platí

$$\lambda_{rs} = \frac{P_s \rho_s 0_r - P_r \rho_r 0_s}{P_r \rho_r 0_s - P_s \rho_s 0_r} ,$$

kde indexy r a s nabývají hodnot FC, FI a IC. Skutečně omezující periodu je nutno určit vyšetřením hranic a určením maxima z poměrů kalendářového a prováděcího času pro každou periodu.

4.2 Správa projektu

Subsystémy by pro analýzu spolehlivosti měly být dostatečně rozsáhlé (aspoň cca 5 000 řádků - kvůli počtu chyb) a práce by se měl zúčastnit dostatečný počet lidí (kvůli odhadu kalendářového času). Malé systémy vedou k příliš širokým intervalům spolehlivosti, malý počet zúčastněných osob k odchylkám od průměrných hodnot použití zdrojů a tedy k chybnému odhadu kalendářového času.

Okamžitá intenzita poruch normálně klesá. Ostrý nárůst představuje nově zavlečené chyby nebo nové prostředí (což naznačuje možnost neadekvátního plánování testů). Příliš pomalý pokles může znamenat buď příliš mnoho přetrvávajících chyb nebo zavlékání přibližně stejného množství chyb při opravách.

5 Závěr

Význam aplikace měření a/nebo predikce spolehlivosti software z různých hledisek lze shrnout do následující tabulky:

Fáze vývoje	Aplikace	Hodnota aplikace	Kvalita výsledků
systémové inženýrství	studie realizovatelnosti	velmi vysoká	slušná -
	vliv kompromisů	vysoká +	slušná +
	nákladové studie	vysoká	slušná
	termínové studie	vysoká +	slušná
	rozvržení spolehlivosti	vysoká -	slušná +
	specifikace spolehlivosti	vysoká -	slušná
testování	sledování stavu	vysoká	dobrá +
	odhad dokončení projektu	vysoká	dobrá -
	studie přepracování projektu	vysoká	dobrá
	zkoumání alternativ řízení	vysoká	dobrá
provoz	předvedení spolehlivosti	vysoká -	vynikající
	řízení změn	vysoká	vynikající
	řízení preventivní údržby	vysoká -	vynikající
všechny	hodnocení technologií	vysoká -	vynikající

Nejdůležitější a nejslibnější oblastí je zřejmě systémové inženýrství a pak provozní část. Možnosti aplikace by neměly být podceňovány, ale ani přeceňovány. Zhruba 2/3 nákladů se objeví až po uvolnění produktu a testy systému stojí cca 40-50% nákladů před uvolněním. Z toho je zřejmé, že asi 80% nákladů máme na začátku testů ještě před sebou.

Literatura

[Be84] Beizer, Boris: Software System Testing and Quality Assurance
Van Nostrad Reinhold Company, New York, 1984
ISBN 0-442-21306-9

[Ma89] Matsumoto Yoshihiro (Editor):
Japanese Perspectives in Software Engineering
Addison-Wesley, 1989
ISBN 0-201-41629-8

[Mu87] Musa, John D.; Iannino, Anthony; Okumoto, Kazuhira:
Software Reliability: Measurement, Prediction, Application
McGraw-Hill, New York, 1987
ISBN 0-07-044093-X

Autor: Ing. Jan Chochola
DemSys - konsorcium
M. Kopeckého 675
708 00 Ostrava - Poruba
tel/fax: (069) 44 24 15