مقایسه مدل های ارزیابی قابلیت اتکای نرم افزار

خلاصه

قابلیت اتکا و اطمینان[1] یکی از مهمترین جنبه های کیفی هر نرم افزار به شمار میرود و تخمین قابلیت اتکای نرم افزار یکی از مسائل دشواری است که نیازمند دقت بالایی در مواجهه با آن است. با این وجود، برای مدیریت کیفیت نرم افزار و انجام رویه های متداول در یک سازمان، مهم است تا جای ممکن به درک صحیحی از قابلیت اتکای دقیق برسیم. در مقاله پیش رو چندین اصل و تکنیک برای تخمین قابلیت اتکا در نرم افزارها ارائه شده است، که از تعریف مفاهیم ضمنی تعیین کننده کیفیت نرم افزار کار را آغاز میکند. تخمین بخشی از نرم افزار نیز مد نظر قرار گرفته است. هدف مفروض تخمین قابلیت اتکا تشکیل یافته از آنالیز ریسک و قابلیت اتکای سیستم های مبتنی بر نرم افزار است. چنین انتظار میرود یک نظر مستند از متخصصان در رابطه با نرم افزار وجود داشته باشد و بروز رسانی بر تخمین قابلیت اتکای تعریف شده با اطلاعات موجود در رکوردهای داده عملیاتی صورت گیرد.

کلمات کلیدی: قابلیت اتکای نرم افزار، مدل نرم افزاری، تخمین، باگ/ خطا، ایراد

1. معرفی

اگر یک ایراد در قسمت سخت افزاری نیازمند زمان برای تعویض عناصر مشکل آفرین باشد، همین مساله را نمیتوان به حوزه نرم افزاری بسط داد، ایراد بوجود آمده در این حوزه نشان دهنده یک خطای پیشین در برنامه است. ایرادهای

[1] Reliability

نرم افزاری به عنوان رویدادهای تصادفی در نظر گرفته میشوند که در یک محور زمانی روی میدهند، آن هم به خاطر کمبود قطعیت در مورد زمان دقیق رخ دادنشان، همین دلیل ارجاعی به مفهوم قابلیت اتکا است، نه کیفیت نرم افزار. یک سیستم نرم افزاری از این نقطه نظر شبیه به بروز خطا در سیستم سخت افزاری به همراه روند بازیابی بوده و شاخص های قابلیت اتکای مشابهی نیز با آن دارد.

با استناد به استاندارد IEEE (ANSI) 982.2 سال 1988 (IEEE 1988)، عناصر الزامی در تعریف نرم افزار از قرار زیر هستند:

خطا. یک خطای انسانی که باعث تولید برنامه نادرست میشود. برای مثال حذف یک وظیفه[2] حیاتی؛

باگ یا ایراد. این مورد نتیجه خطای انسانی است و نشان دهنده حادثه داخلی است که باعث میشود سیستم مورد انتظار عمل نکند.

در زبان حال حاضر، واژه خطا هم برای اشتباه کردن (ایراد[3])، و هم برای تاثیر مستقیم اش در برنامه (خطا یا باگ[4]) به کار میرود.

2. معیار برای آنالیز مدل های قابلیت اتکای نرم افزار

اکثر مدلهای کنونی ارتقای قابلیت اتکای نرم افزار زمان را به عنوان یک متغیر پیوسته در نظر میگیرند (چه زمان تقویمی، ساعت یا زمان اجرا). با این وجود سیستم هایی وجود دارند که مانند سیستم های پرداشگر تراکنش ها، که در آنها قابلیت اتکا باید متناسب با تراکنش های موفق انجام شده در نظر گرفته شود. علاوه بر این، سیستم های دیگری هم هستند که مانند نرم افزار کنترل موشکی (سفینه های فضایی) که در آن دسته قابلیت اتکا میتواند تعداد دفعات موفق پرتاب موشک باشد. سیستم های از این دست نیازمند رویکرد محتاطانه ای در قبال زمان هستند، به عبارت دیگر تعداد دفعات اجرای نرم افزار.

---

[2] task
[3] error
[4] Fault – bug

شاخص «پوشش کد» (بخش واقعی از کد که تست شده است) در فاز تست کردن میتواند تخمین قابلیت اتکای نرم افزار را به طرز چشم گیری افزایش دهد: تست کردن میتواند به اشباع برسد که در این شرایط بخش های جدید کد تست نمیشوند. پس تخمین قابلیت اتکا که بیشتر بر زمان اجرا متکی است میتواند قابلیت اتکای برنامه را بیش از حد واقعی تخمین بزند.

راه های متعددی برای ارزیابی ارزش یک مدل نرم افزاری وجود دارد:

اعتبار قابل پیش بینی. این مورد نشان دهنده ظرفیت مدل برای پیش بینی است، با توجه به رفتار شکست آتی، که هم میتواند در طول فاز تست کردن و یا فاز اجرا صورت گیرد. پیش بینی های بدست آمده از این رفتار شکست و فاز متناسب پیشین حاصل میشوند. این جنبه را میتوان به بخش های بیشتری نیز تقسیم کرد:

دقت (سطح دقت) که از شباهت محتاطانه اندازه گیری میشود

انحراف۵ (شیب)، که از روی U-دیاگرام اندازه گیری میشود

گرایش۶، یا تغییر سیستماتیک شیب از مقادیر کوچک به سمت مقادیر بزرگ در زمان شکست، که از روی Y-گرافیک اندازه گیری میشود

نویز، از روی تغییر نسبی روی داده در نرخ پیش بینی خطا حاصل میشود

توانایی. ظرفیت مدل برای تخمین با دقت مطلوب که مدیران، مهندسان و کاربران نرم افزار برای برنامه ریزی و مدیریت پروژه های توسعه نرم افزار به آن نیاز دارند، یا کنترل تغییرات روی داده در سیستم های عملیاتی نرم افزار. این ابعاد شامل قابلیت اتکای کنونی، موعد پیش بینی شده برای رسیدن به قابلیت اتکای هدف و همچنین هزینه رسیدن به هدف میشوند.

کیفیت برانگاشت. اگر یک برانگاشت (فرضیه) ایجاد شده توسط یک مدل میتواند مورد تست قرار گیرد، پس این جنبه به میزانی اشاره دارد که مدل توسط داده واقعی مقبول بوده است: هر چند، تست کردن برانگاشت ممکن نیست، جنبه

[5] Inclination
[6] Tendency

ها به معقول بودن از نظر جامعیت منطقی اشاره دارند و تجربه بدست آمده در مهندسی نرم افزار. همچنین، شفافیت و توصیف یک برانگاشت مدل باید مورد بررسی بیشتر قرار گیرد.

قابلیت به کار گیری به مفید بودن مدل داخل چارچوب محصولات مختلف نرم افزاری (اندازه، ساختار، توابع) اشاره دارد، محیط گسترده تولید، محیط های عملیاتی گوناگون و فازهای مختلف چرخه حیات.

سادگی. تشکیل یافته از سهولت در جمع آوری داده است که برای شخصی سازی بیشتر مدل ضروری تلقی میشود. همچنین به سادگی مفهومی نیز اشاره دارد که در آن ناظران مدل (مهندسان نرم افزار، مدیران پروژه و متخصصان قابلیت اتکا و سایرین) میتواند طبیعت مدل و برانگاشت آن را درک کنند و مورد استفاده اش برای یک مشکل خاص را تعیین نمایند و همچنین میزانی که در آن مدل در کاربرد مشخص اش از واقعیت جدا میشود. جدا از این مورد، سادگی شامل سادگی در پیاده سازی نیز میشود، به طریقی که مدل میتواند ابزاری کارآمد در مدیریت و مهندسی نرم افزار شود.

سهولت در اندازه گیری پارامترها. این جنبه تعداد پارامترهای مورد نیاز مدل و درجه سختی مرتبط در تخمین شان را لحاظ میکند.

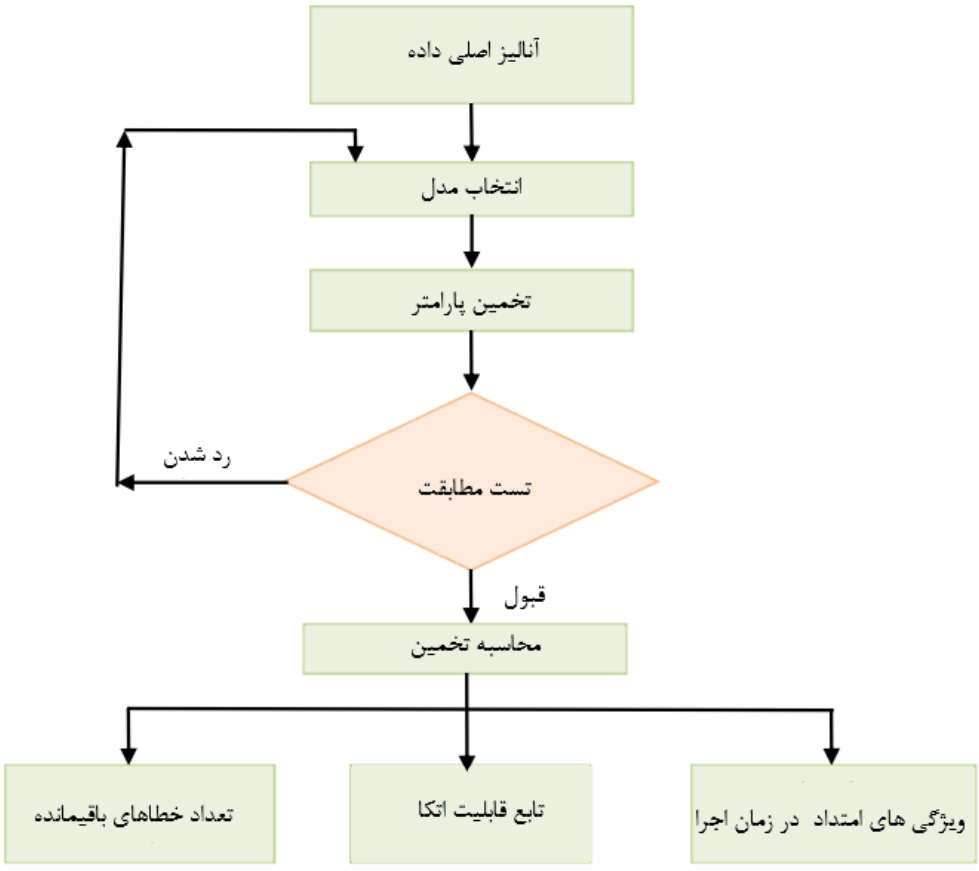
حساس نبودن به نویز. جنبه ای که به ظرفیت مدل جهت انجام دقیق پیش بینی ها اشاره دارد، حتی زمانی که داده شکست ناقص یا حاوی ابهام باشد.

گوئل (گوئل 1985) پیشنهاد میکند پیش از اعمال یک مدل، خطا ها باید به صورت مستقل ارزیابی شوند، بسته به زمان تقویمی، زمان اجرا یا دفعات اجرای برنامه.

گوئل (گوئل 1985) عقیده دارد پیش از اعمال یک مدل، ارزیابی خطاهایش باید مورد مطالعه قرار گیرند، با استناد به زمان تقویمی، اجرا و یا دفعات اجرا. پس متغیرهای مهمی را میتوان انتخاب کرد و گرایش بدست آمده برای یک کلاس خاص از مدل نیز از همین راه است. پس از تخمین، پارامترهای مدل در فاز اول انتخاب میشود، یک تست مطابقت مبتنی بر مدل داده اجرا میشود (تصویر 1). اگر مدل سازگار باشد، معیارهای تخمین گر هم محاسبه خواهند شد و

تصمیمات مرتبط با تکمیل یا تداوم تست کردن اتخاذ میشود. طبق تصویر 1، گام تعیین کننده در مورد اعمال مدل سنخیت آن است و نه مقایسه بین مقادیر تخمین زده شده و واقعی.

تکنیک های موجود برای ارزیابی یک مدل در این بخش مورد بررسی قرار گرفته اند، در چارت-U فاکتور بایس[7] و گزارش شباهت محتاطانه تنها زمانی به کار گرفته میشوند که نمونه های رویداد وجود داشته باشند. در مورد نارسایی های نرم افزار باید گفت حداقل برخی از ایرادات نرم افزاری باید رخ داده و مورد بررسی قرار گیرند. در مورد دستگاه های برنامه پذیر قابل اتکا در حال کار نیز باید گفت این فرض کمی غیر واقعی خواهد بود. پس این روش ها بیشتر برای مدل های در حال رشد قابلیت اتکای نرم افزار کارآیی دارند، که به طور عمده مرحله پیشبرد چرخه حیات یک نرم افزار را نشان میدهد.



تصویر 1. اعمال یک مدل برنامه های قابل اتکا

[7] Bayes

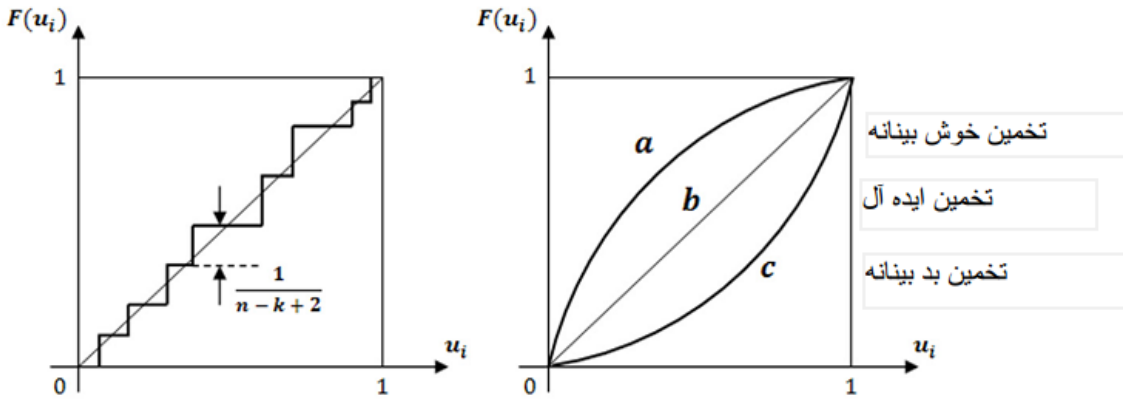
2.1. چارت-U

روش های دیگری هم برای اعتبارسنجی و مقایسه مدل ها وجود دارد، از جمله شباهت اعتبار-متقابل (بائت 1998). چارت-U برای سنجش نزدیک بوده مقدار تابع تعیین توزیع تجمعی بدیهی به مقدار توزیع واقعی به کار میرود (بر اساس مشاهدات موجود). مشخص است که متغیر تصادفی توزیع یکنواخت در بازه های مختلف دارد. پس اگر دستاوردها (برای مثال زمان شکست) مشاهده و محاسبه گردد، پس باید یک دستاورد از متغیر تصادفی واحد باشد. هرگونه انحراف از این یکنواخت بودن نشان دهنده انحراف خواهد بود.

برای پیدا کردن انحرافات (در صورت وجود) تابع توزیع نمونه از مشاهدات تبدیل شده به کار میرود. نمودار یک تابع پلکانی است که از تعداد بازه ها تشکیل یافته است. پس یک تابع پلکانی صعودی است که در هر گام بر روی یک مقدار از بعد افقی نوشته میشود.

هر چقدر این نمودار به خط شیب نزدیک باشد، نزدیکتر خواهد بود. در طرف دیگر، هرگونه انحراف سیستماتیک از شیب یکنواخت نشان دهنده ابهام در توزیع احتمال است.

این مفاهیم را میتوان به عنوان معیار عملیاتی نیز در نظر گرفت، آنهم از طریق یافتن فاصله کولموگروف (انحراف عمودی بیشینه دقیق) بین خط پیش بینی دقیق از منحنی 1 و دیاگرام متاثیر از آن.



تصویر 2. چارت-U، گسسته (چپ) و پیوسته (راست)

2.2. چارت-۷

چارت-Y درجه وابستگی مدل گرایش را نشان میدهد؛ برای مثال یک مدل میتواند در مورد تعداد نارسایی های نرم افزار در ابتدا بدبین باشد، بعضی اوقات بسیار خوشبین.

این مساله در نتیجه دنباله نگاشت $u_i = \hat{F}(t_i)$ است که در بخش قبل به صورت زیر تعریف شد؛

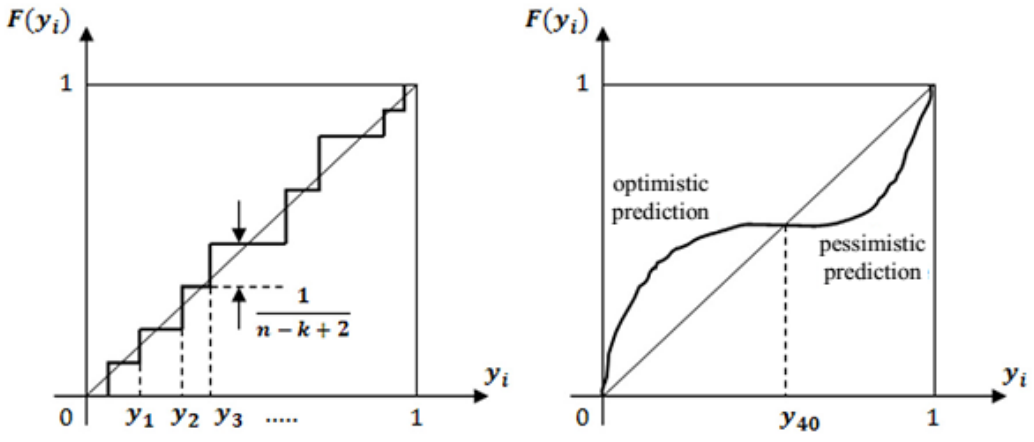
$$x_i = -\ln(1-u_i) \qquad (1)$$

و

$$y_i = \frac{\sum_{j=1}^{i} x_j}{\sum_{j=1}^{M} x_j}, \qquad (2)$$

که در آن $i \le M$ است.

اینکار را میتوان با یک معیار عملیاتی توسط محاسبه فاصله کولموگروف نیز انجام داد؛ $\max_i \left(|x_i - y_i|\right)$ ، بین متغیرهایی که بالا تعریف شده اند.



تصویر 3. چارت-U، گسسته (چپ) و پیوسته (راست)

2.3. فاکتور بایس

فاکتور بایس نشان دهنده معیار مقایسه رسمی از مدل های بایسیان است. دو مدل رقابتی H_1 و H_2، به ترتیب وجود دارند. فاکتور بایس تشکیل یافته از گزارش بین محدوده احتمالات دو مدل مقایسه شده به صورت زیر است:

$$BF(H_1,H_2)=\frac{p(t_1,...,t_m\mid H_1)}{p(t_1,...,t_m\mid H_2)}\ , \qquad (3)$$

که در آن

$$p(t_1,...,t_m\mid H_i)=\int p(t_1,...,t_m\mid \theta_i,H_i)\cdot p(\theta_i\mid H_i)\cdot d\theta_i\ . \qquad (4)$$

محاسبه فاکتور بایس نیازمند منابع محاسباتی خواهد بود. برای تجدید نظر در تخمین محدوده احتمالات مبتنی بر چارتهای توزیع پسینی، لطفا به ارجاع (دی سیسیو 1997) مراجعه کند.

2.4. شباهت محتاطانه و نرخ شباهت محتاطانه

شباهت محتاطانه درجه دقت یک مدل را نشان میدهد (که با $f_A(t)$ در تابع چگالی احتمال داده، بدست آمده از مدل A حاصل میشود. علاوه بر این $t_1,...,t_m$ های بدست امده از بروز نارسایی ها هم وجود دارند.

مدلی از شباهت محتاطانه به صورت زیر قابل توصیف است:

$$PL_A=\prod_{j=1}^{M} f_A(t_j) \qquad (5)$$

محصول فاکتورها معمولا نزدیک به صفر است و شاخصی واضحتر از لگاریتم شباهت محتاطانه حاصل میشود.

گزارش شباهت محتاطانه ظرفیت دو مدل را برای پیش بینی مجموعه مشخصی از داده مقایسه میکند. با $f_A(t)$ و $f_B(t)$ نمایش داده میشوند و نشان دهنده تابع چگالی احتمال داده بدست آمده، به ترتیب از مدل A و B هستند.

گزارش شباهت محتاطانه PLR_i^{AB} به صورت زیر تعریف میشود:

$$PLR_i^{AB}=\prod_{j=1}^{i}\frac{f_A(t_j)}{f_B(t_j)} \qquad (6)$$

این نرخ باید در صورت برتری مدل A بر مدل B افزایش یابد و بر عکس، کاهش یابد.

به سادگی میتوان دید نرخ شباهت محتاطانه نسخه ای ساده شده از فاکتور بایس است: اگر مشاهدات مستقل هستند و دامنه احتمال گسسته، پس با هم مصادف خواهند شد. به هر حال در اغلب موارد نرخ شباهت محتاطانه به سادگی قابل محاسبه است.

3. نتیجه گیری

در مقالات مدل های بسیاری پیشنهاد شده اند، اما هر یک مزایا و معایب خود را دارند، بعضی از آنها در اکثر موارد متداول تر هستند. یکی از مدلهای بازبینی شده اجازه انجام آنالیز بر اساس داده نارسایی ناموجود را نمیدهد. برای مثال تاریخچه استفاده از نرم افزار در حالت عملیاتی اش در یک بازه از زمان بدون هیچ خطا / نارسایی. مشکل متداول دیگر این مدل ها این است که تنها از پیش نیازهای قابلیت اتکای متوسط پشتیبانی میکنند. هیچ راه حلی برای این مشکل وجود ندارد.

مدل های کنونی قابلیت اتکا پیچیدگی کاربردی یا درجه پوشش تست را مد نظر قرار نمیدهند.

با در نظر گرفتن این موارد، اوضاع باز هم پیچیده میشود، چرا که نرم افزار تحت بررسی خود به خود اجرا نمیشود و بخشی از یک سیستم است، متشکل از سخت افزار، سیستم عامل، واسط (برای مثال درایورهای دستگاه ها و واسط های ارتباطاتی)، و احتمالا برنامه ها (برای مثال محیط های مجازی).

به همین ترتیب اکثر مدل های بررسی شده طوری طراحی شده اند تا افزایش عناصر قابلیت اتکا را مدل کنند. این مساله زمانی مهم است که برنامه توسط تیم منضبط تولید شده باشد و گزارشات در مورد نارسایی های هر مرحله از توسعه، یا حداقل، آمار نارسایی ها در دسترس باشد. با این وجود، از نقطه نظر کاربر نرم افزار، واقع بینانه تر است اگر فرض شود تنها رکوردهای داده عملیاتی در دسترس باشد.

چند پیشنهاد در رابطه با کاربرد مدل های قابلیت اتکای نرم افزارها میتواند ارائه کرد:

مدل هایی که معماری نرم افزار، پیچیدگی نرم افزار، درجه پوشش تست، اعتبار سنجی و مطابقت ارزش ها در کنار نظرات نظام مند متخصصان را در اولویت قرار میدهند باید ارجحیت بیشتری داشته باشند.

در برنامه هایی که نیازمند درجه استقلال بالایی هستند، مدل های قابلیت اتکای برنامه باید تنها در تناسب با سایر مدل ها به کار روند. آن دسته از مدل هایی که سطح کیفی مطلوبی دارند، در غیر اینصورت ابعاد تست به طرز اغراق شده ای افزایش می یابد. در بین این روش ها، روش های رسمی، بازرسی های نرم افزاری و بازبینی ها، آنالیز کد-استاتیک، تست کردن سیستماتیک نرم افزار و غیره میتواند بخشی از فرآیند باشد.

شاید چنین به نظر برسد که مدل های بایس دورنمای بهتری از مدل های سنتی ارائه میکنند. یکی از مزیت های رویکرد بایس این است که این اجازه را میدهد تا جامعیت انواع مختلف اطلاعات، حتی تصمیمات انسانی نیز معنی پیدا کنند.

هر توسعه دهنده نرم افزاری نباید صرفا بر روی یک مدل دقیق شود، بلکه مجموعه ای از مدل ها را انتخاب کند و در انها نتایج شان را به هر نحو در انتها ترکیب نماید.

References

Basu S. and Ebrahimi N., “Estimating the Number of Undetected Errors: Bayesian Model Selection” in Proceedings of the The Ninth International Symposium on Software Reliability Engineering, 1998, p. 22;

Brocklehurst S. and Littlewood B., “New Ways to Get Accurate Reliability Measures” IEEE Software Transaction, vol. 9, no. 4, pp. 34-42, 1992;

DiCiccio T. J., Kass R. E., Raftery A., and Wasserman L., “Computing Bayes Factors by Combining Simulation and Asymptotic Approximations” Journal of the American Statistical Association, vol. 92, no. 439, pp. 903-915, 1997;

Gelman A. B., Carlin J. S., Stern H. S., and Rubin D. B., „Bayesian data analysis”, Boca Raton: Chapman and Hall, 2000;

Goel A. L., “Software Reliability Models: Assumptions, Limitations, and Applicability” IEEE Transactions on Software Engineering, vol. 11, no. 12, pp. 1411-1423, 1985;

Iannino A., Musa J. D., Okumoto K., and Littlewood B., “Criteria for software reliability model comparisons” ACM SIGSOFT Software Engineering Notes, vol. 8, no. 3, pp. 12-16, 1983;

Lyu M. R. and Nikora A., “Applying Reliability Models More Effectively” IEEE Software Transactions., vol. 9, no. 4, pp. 43-52, 1992;

Mihalache A., „Când calculatoarele greesc: fiabilitatea sistemelor de programe (software)”, Bucureti: Editura didactic i pedagogic, 1995;

Whittaker J. A. and Voas J., “Toward a More Reliable Theory of Software Reliability” Computer, vol. 33, no. 12, pp. 36-42, 2000; IEEE (ANSI) Standard 982.2/1988. Software Reliability Terminology.