Suatu Tinjauan Statistik Mengenai Reliabilitas Perangkat Lunak

Oleh :
Hendarman *)

Abstract

Failure times of software undergoing random debugging can be modelled as order statistics of independent but nonidentically distributed exponential random variables. Using this model inferences can be made about current reliability and, if debugging continues, future reliability. This model also shows the difficulty inherent in statistical verification of very highly reliable software such as that used by digital avionics in commercial aircraft.

Intisari

Dengan cara 'debug' acak, banyaknya kegagalan perangkat lunak dalam suatu saat dapat disajikan dalam model perubah acak berdistribusi eksponensial yang berorder statistik bebas tetapi tidak identik. Dengan menggunakan model ini, maka beberapa kesimpulan mengenai reabilitas perangkat lunak disaat itu dapat diambil. Dan jika debugging dilanjutkan, juga reliabilitas selanjutnya.Model ini menunjukan kesulitan yang terjadi dalam verifikasi statistik dari perangkat lunak dengan reliabilitas sangat tinggi, seperti yang digunakan dengan avionik digital dalam penerbangan komersil.

1. PENDAHULUAN

Makalah ini membahas beberapa aspek analisis statistik mengenai reliabilitas perangkat lunak. Tujuannya adalah mengungkapkan beberapa masalah dan kesulitan yang berkenaan dengan reabilitas perangkat lunak, dan bukan mengemukakan jalan keluar atau solusi ataupun metode baru untuk menangani masalah tersebut. Tinjauan pertama adalah hal-hal yang berkaitan dengan reliabilitas tingkat menengah dan sekelompok model, khususnya model-model statistik order eksponensial (Exponential Order Statsitics) yang, dalam hal ini, sangat berguna untuk pengembangan selanjutnya. Tingkat-tingkat reliabilitas menengah ini ditentukan oleh adanya kemungkinan dapat dilakukannya pengujian dalam waktu yang paling sedikit lebih lama dari rata-rata waktu yang terpakai antara kegagalan-kegagalan. Kedua adalah reliabilitas tingkat tinggi (ultrahigh-reliability), misalnya, jika banyaknya kega- galan dibawah satu per satu milyar misi. Dalam hal ini biasanya tenggang waktu, bahkan untuk mendekati rata-rata waktu, yang dibutuhkan antara kegagalan-kegagalan hampir tidak mungkin dapat diuji. Selain itu, juga hampir tidak dimungkinkan untuk mengembang- kan prosedur pengujian yang cukup mewakili permasa- lahan yang dibahas.

Disaat suatu perangkat lunak mengoperasikan suatu fungsi, ada dua kemungkinan bahwa perangkat lunak tersebut menjalankannya dengan baik, seperti apa yang diharapkan, atau tidak. Setelah dikembangkan denganterampil dan hati-hati, dimungkinkan perangkat lunak itu dapat bekerja dengan baik setiap saat digunakan. Tetapi,selalu ada kemungkinan bahwa suatu saat (mungkin ketika mengoperasikan program, yang cukup rumit dan panjang) perangkat lunak itu tidak bekerja seperti apa yang diharapkan; sehingga ada suatu ketidak- pastian tentang sejauhmana perangkat lunak itu dapat bekerja dengan baik. Dalam makalah ini, dengan melakukan suatu studi banding dari sumber-sumber informasi yang ada, terlebih dahulu akan dicoba untuk menterjemahkan segala sesuatu yang telah diketahui tentang perangkat lunak kedalam beberapa bentuk estimasi tentang bagaimana perangkat lunak tersebut bekerja. Kemudian, suatu pendekatan mengenai ketidakpastian dapat dilakukan dengan cara membang- un suatu model dengan menggunakan teori kemung- kinan dan metode-metode statistik agar dapat diambil suatu inferensi (kesimpulan) dari permasalahan yang ada. Dengan demikian, inti pembahasan makalah ini dapat dikatakan sebagai suatu prediksi mengenai reliabilitas perangkat lunak.

Adalah sangat sulit untuk mengembangkan suatu prosedur pengujian yang cukup mewakili permasalahan yang terjadi di lapangan. Oleh karena itu, untuk tujuan analisis statistik mengenai reliabilitas perangkat lunak ini, biasanya banyak informasi yang diabaikan dan ukuran kinerjanya seringkali diperkecil kedalam suatu rate kegagalan tertentu. Metode statistik merupakan salah satu cara untuk mengestimasi rate kegagalan selanjutnya dari

*) Staf Peneliti Bidang Teknologi Peluncuran dan Operasi Antariksa, LAPAN.

semua kegagalan yang diselidiki (yang mungkin terjadi pada saat pengembangan, debugging, pengujian, ataupun penerapan lapangan). Dalam makalah ini akan ditunjukkan suatu model yang sangat dise- derhanakan dari permasalahan yang ada, tetapi secara statistik tetap sangat berguna jika data mengenai adanya kegagalan yang cukup banyak dan bervariasi dapat dikumpulkan.

2. SUATU MODEL KEGAGALAN PERANGKAT LUNAK.

Perhatikan suatu model dari bugs dan penggunaan perangkat lunak yang akan mengarah pada suatu penyelidikan mengenai waktu-waktu kegagalan dengan perubah acak berdistribusi eksponensial yang berorde statistik bebas dan tidak identik. Misalkan perangkat lunak tersebut beroperasi dengan menerima data dari suatu ruang input dan mentransformasikannya ke dalam data output yang bisa benar atau salah. Hal tersebut dilakukan pada sederet input yang diambil secara acak dan bebas dari suatu ruang input yang mungkin berdasarkan suatu distribusi tertentu. Anggap bahwa kondisi internal dari komputer adalah identik untuk setiap input. Jika F, suatu himpunan bagian dari ruang input yang berkorespondensi dengan input-input yang mengakibatkan bug dalam perangkat lunak dan menghasilkan suatu output yang salah. Misalkan distribusi input tersebut memberikan peluang p pada F, maka dalam skenario di atas bug ini akan mempunyai waktu-waktu antar-kegagalan yang berdistribusi geometri dengan rata-rata 1/p. Jika p kecil, maka distribusi waktu antar-kegagalan tersebut dapat didekati dengan suatu distribusi eksponensial. Jika suatu bug dihilangkan (fix sempurna) saat mana bug tersebut memanifestasikan sendiri dalam suatu output yang salah, maka akan diperoleh suatu waktu tunda yang berdistribusi eksponensial tunggal hingga manifestasi tersebut berakhir.

Selanjutnya, asumsikan bahwa peluang kejadian simultan lebih dari satu bug dari semua input diabaikan. Hal ini memungkinkan kita untuk membuat model dari waktu-waktu manifestasi yang terpisah sebagai perubah-perubah acak yang bebas dan kontinu.

Model Statistik Orde Eksponensial (EOS) dapat diuraikan sebagai berikut :

misalkan $0 <= T_1 <= T_2 <= T_3 <= \ldots <= T_j <= \ldots$

perubah-perubah acak yang berkorespondensi dengan waktu-waktu manifestasi dari bug dalam suatu perangkat lunak. Setiap bug mempunyai rate kegagalan yang berkaitan dengan waktu-waktu manifestasi tersebut. Jika semua bug diberi indeks secara sembarang, maka, misalkan, λ_i merupakan rate kegagalan dari bug ke I.

Misalkan X_i sama dengan waktu kemunculan bug ke i

$$P\{X_i > t\} = \exp(-\lambda_i t) \qquad (2\text{-}1)$$

Perubah-perubah acak $\{ X_i \mid i = 1,2,3, \ldots \}$ satu sama lain bebas dan orde statistiknya dapat dinyatakan sebagai : $\{T_j \mid j = 1,2,3, \ldots \}$. Informasi lebih jauh tentang model-model EOS dapat di lihat di Miller [2] dan Scholz [4].

Hal yang perlu diperhatikan dengan asumsi-asumsi yang dibuat dalam model EOS antara lain ;

a) Input-input diambil secara bebas dari sebuah distribusi yang digunakan; yakni, input-input yang terdistribusi secara bebas dan identik (independent identically distributed / i.i.d)
b) Kondisi internal daripada mesin komputer adalah identik untuk setiap input.
c) Bug-bug ditentukan dengan baik.
d) Tidak ada interaksi bug.

Selain itu, perlu dicatat pula bahwa disini tidak ada asumsi-asumsi ditujukan pada rate kegagalan λ_1, λ_2, λ_3,... , λ_i, ... Nilai-nilai λ ini bisa terhingga ataupun tak terhingga banyaknya, tetapi

$$\sum_{i=1}^{\infty} \lambda_i < \infty \qquad (2\text{-}2)$$

Disini, perlu sekali mempertimbangkan inferensi yang terkait dalam konteks model-model statistik orde eksponensial tersebut. Sebab, langkah ini biasanya dapat mengungkapkan beberapa kesulitan yang terkandung didalamnya. Beberapa masalah yang mungkin timbul antara lain;

a) Untuk mengambil inferensi tentang bagaimana suatu perangkat lunak akan bekerja dalam bidang yang bersangkutan, kita harus tahu daerah definisi dari input serta ditribusi input yang akan timbul.

b) Asumsi-asumsi dari input i.i.d. untuk serangkaian eksekusi mungkin dapat memenuhi beberapa jenis batch tertentu pada program-program aplikasinya, tetapi tidak pada perangkat lunak lain seperti perangkat lunak sistem-sistem ataupun 'real timecontrol software'.

c) Model-model ini mencakup masalah-masalah identifikasi bug-bug yang terpisah, fixes tak sempurna, ketergantungan antar bug dan faktor-faktor lain yang cenderung membuat data semerawut.

d) Banyaknya model-model yang dimungkinkan dan beberapa sifat yang tak diharapkan dari fungsi-fungsi yang monoton membuat prediksi yang akurat semakin sulit. (Lihat gambar 4 serta bahasannya

3. PREDIKSI RELIABILITAS

Masalah prediksi reliabilitas merupakan sesuatu yang khusus dalam pemeliharaan perangkat lunak. Suatu estimasi tentang jumlah bug baru yang akan muncul selama suatu interval waktu yang akan datang akan sangat berguna dalam hal perancangan. Dengan menggunakan paradigma EOS, estimasi ini merupakan masalah yang cukup sulit dikarenakan dua intensitas yang monoton akan saling bersesuaian dalam suatu interval terhingga, kemudian berbeda (diverge) dalam suatu interval selanjutnya. Kedua intensitas tersebut bisa sangat berdekatan sehingga hampir tak mungkin untuk dapat menentukan data yang mana yang paling memenuhi. Hal ini dapat dilihat dari Gambar 1 dan Gambar 2;

Gambar 1 menunjukkan beberapa data kegagalan; yakni, jumlah bug yang dimanifestasikan selama waktu t dengan $0 <= t <= 100$.

Gambar 2 menunjukkan dua model yang mungkin, yakni,

- garis lurus dengan persamaan : $M(t) = t/2$ dan
- garis lengkung dengan persamaan :

$$M(t) = 47 \log(0{,}01\, t + 1) / \log 2$$

Hal ini menunjukan jumlah kejadian yang muncul dalam Proses Poison Tak Homogen (PPTH); Garis lurus merupakan suatu Proses Poison Homogen (PPH) sedangkan Garis lengkung merupakan suatu model Musa-Okumoto [3]. Proses Poison Homogen adalah suatu masalah yang terbatas dalam paradigma EOS. sedangkan model Musa-Okumoto merupakan suatu model EOS stokastik ganda. Jadi berdasarkan paradigma EOS, kedua model tersebut haruslah dipertimbangkan.

Dalam Gambar 3, data digabungkan dalam dua fungsi rata-rata. Disini terlihat bahwa kesalahan acak (noise) dalam data adalah sama dengan atau lebih besar dari pada selisih dari kedua model itu. Melalui uji hipotesa, kedua model tersebut dimungkinkan sebagai model yang benar.

Gambar 4 menunjukan prediksi dari rangkuman model-model untuk yang akan datang. Kedua model dibedakan oleh suatu faktor yang medekati 2 dalam jumlah kejadian yang diharapkan dalam interval [100,200].

Dari serangkaian gambar-gambar ini, terlihat bahwa suatu keadaan acak dan kemungkinan ketidak-tepatan akan terjadi ketika jumlah bug yang akan datang diprediksi dari jumlah bug dari kejadian-kejadian yang telah lalu. Datanya dapat dihasilkan dengan menggunakan kalkulator dari Proses Poison Homogen dengan $M(t) = t/2$. Kelihatannya, ada sejumlah batasan terhadap inferensi yang dapat dibuat tanpa informasi lain yang akan membatasi sekelompok model-model yang dapat diterima.

Model-model pertumbuhan reliabilitas telah banyak digunakan dan berhasil dengan baik untuk kasus-kasus dengan tingkat reliabilitas menengah (moderat). Sebagai contoh, Currit, Dyer dan Mills [1] telah berhasil menggunakan model-model demikian untuk suatu sistem dimana mereka ambil rate kegagalan dengan kasar dalam lingkungan satu kegagalan per 5000 uji kasus. Akan tetapi, untuk mendapatkan tingkat reliabilitas sangat tinggi melalui pertumbuhan reliabilitas diperlu- kan pengujian yang sangat panjang. Beberapa penomena dapat dilihat sebagai berikut; Untuk data yang digunakan oleh Musa-Okumoto , reliabilitas terakhir (yang diukur oleh waktu terakhir antara kegagalan) yang terbaik adalah dalam lingkungan (neighborhood) 1/100 dari total waktu uji dan untuk tingkat reliabilitas yang lebih tinggi dapat diperoleh dalam lingkungan yang kurang dari 1/1000. Jenis data ini, dimana jumlah kegagalan yang wajar diukur dan programnya bukan reliabilitas tingkat tinggi, merupakan domain yang cocok untuk menerapkan model-model reliabilitas perangkat lunak secara statistik. Kita akan tetap mendapatkan suatu nilai jika estimasi-estimasi yang kita buat sangat tidak tepat atau kita tidak mempunyai suatu tingkat kepercayaan yang sangat tinggi. Jika demikian masalahnya, maka model-model dengan asumsi-asumsi yang salah dapat digunakan; dan pedekatan yang dihasilkan dapat diterima dan tetap berguna.

Dalam masalah-masalah yang ditunjukan di atas, pembuatan inferensi tentang kinerja dalam waktu yang akan datang berdasarkan kinerja dalam waktu yang lalu dari perangkat lunak itu disertai informasi lainnya. Masih banyak masalah-masalah dalam permodelan dan inferensi dalam pertumbuhan reliabilitas data. Ada tiga macam kinerja utama yang menarik untuk diperhatikan antara lain;

- reliabilitas terakhir,
- jumlah bug yang diharapkan dapat ditemukan dalam suatu interval waktu yang akan datang, dan
- tambahan waktu pem-bug-an untuk mencapai suatu tingkat reliabilitas yang diinginkan.

Dalam membuat inferensi tentang kuantitas ini, bebrapa masalah yang akan muncul antara lain;

a) Menentukan ketelitian titik-titik estimasi, mungkin dengan menggunakan interval-interval kepercayaan.

b) Mengumpulkan lebih banyak informasi kedalam prosedur inferensi
c) Mencari cara-cara untuk menentukan kelas-kelas terbatas dari model-model pertumbuhan reliabilitas.
d) Memperbaiki ketidak-tentuan dalam distribusi bidang yang digunakan.
e) Mempertimbangkan skema-skema sampling yang berbeda.
f) Menghitung pembatasan-pembatasan dari suatu pendekatan inferensi secara statistik.
g) Menangani fixes yang tak sempurna.
h) Memisahkan masalah-masalah dengan akurasi yang diperoleh dari model yang buruk atau dari model yang baik tapi memiliki karakteristik inferensi yang buruk.

Dalam hal ini, tentunya, sangat diperlukan teknik-teknik statistik yang lebih canggih. Teknik-teknik ini dapat digunakan sebagai alat manajemen. Disinilah kesempatan bagi para matematika/statistikawan untuk dapat menyumbangkan pemikirannya. Analisis statistik yang benar dan tepat merupakan hal yang sangat penting untuk suatu perangkat lunak yang mampu mencapai tingkat realibilitas yang sangat tinggi demi alasan-alasan keamanan. Tetapi, hal ini mungkin merupakan masalah lain dari pertumbuhan reliabilitas perangkat lunak yang biasa ditujukan untuk tingkat - tingkat reliabilitas menengah.

4. ANALISIS PERANGKAT LUNAK DENGAN TINGKAT RELIABIILITAS SANGAT TINGGI.

Perangkat lunak yang dugunakan dalam kendali waktu-nyata dan sistem-sistem yang kritis terhadap keamanan haruslah mempunyai tingkat reliabilitas yang sangat tinggi. Perangkat lunak dalam komputer-komputer kendali penerbangan dijital yang ada dalam pesawat-pesawat terbang komersial tentunya sangat kritis terhadap keamanan penerbangan. Reliabilitas pada orde 10^{-9} kegagalan per jam adalah yang diharapkan. Mencoba menghitung tingkat reliabilitas setinggi itu dengan nilai-nilai parameter statistik adalah sangat sulit dan percuma saja. Komisi Teknik Radio untuk Pener- bangan, USA, menanggapi perhitungan reliabilitas ini dalam '*Software Considerations in Airbone Systems and Equipment Certification*', sebagai berikut;

> "Selama persiapan dokumen ini, teknik-teknik untuk mengetimasi peluang verifikasi-lanjut dari kesalahan perangkat lunak telah diuji. Tujuannya adalah untuk mengembangkan prasyarat numerik dalam peluang-peluang seperti itu untuk peralatan berbasis komputer dijital dan verifikasi sistem-sistem. Kesimpulan yang diperoleh adalah bahwa dengan metode terakhir yang tersedia, hasil-hasil yang diperoleh tidak mencapai tingkat kepercayaan yang diperlukan untuk maksud ini. Dengan demikian, dokumen ini tidak menyatakan verifikasi-lanjut dari prasyarat kesalahan perangkat lunaknya".

Dalam '*Advisory Circular*' No. 25.1309-1, FAA menggunakan nilai 10^{-9} untuk menunjukkan "nilai yang sangat tidak mungkin". Kejadian demikian tidak akan mungkin terjadi selama waktu hidup suatu armada pesawat terbang. Untuk memperoleh gambaran yang lebih jelas, perhatikan contoh persoalan berikut ;

Sebuah pesawat terbang yang waktu hidupnya 30 tahun terbang selama 10^4 hari, dengan tenggang waktu terlama 10 jam terbang per hari. Jadi, untuk suatu armada pesawat terbang yang berjumlah 10^3 pesawat, akan terkumpul sejumlah 10^8 jam terbang. Kita berharap 0,10 kejadian dari suatu peristiwa dengan peluang 10^{-9} selama waktu ini. Distribusi Poison merupakan suatu model yang baik untuk peristiwa-peristiwa yang terjadi ini. Dengan demikian, peluang untuk tidak terjadi apa-apa adalah exp(-0,10) = 0,90. Hal ini kelihatannya tidak mungkin.

Dalam teori, verifikasi statistik untuk semua tingkat reliabilitas adalah dimungkinkan. Dalam prakteknya, sedikitnya kita berhadapan dengan tiga masalah/kesulitan utama, antara lain ;

- Distribusi yang digunakan dalam pengujian tidak akan sesuai secara tepat dengan distribusi yang diguna kan di lapangan.
- Kesesuaian mungkin tidak sempurna.
- Waktu pengujian mungkin terbatas.

Distribusi yang digunakan bisa jadi merupakan suatu masalah besar. Distribusi ini memungkinkan membiaskan distribusi pengujian setelah melakukan bug yang para penguji pikir lebih cenderung ada di perangkat lunaknya, tetapi pengetahuan yang lebih mendalam dalam distribusi yang digunakan di lapangan ini sangat diperlukan untuk menghilangkan bias dalam mengesti- masi reliabilitas. Masalah ketidak-sempurnaan fixes (kesesuaian) dapat dihindari dengan mempertimbangkan bahwa perangkat lunak betul-betul baru setelah setiap fix dan, dengan demikian, tidak mencoba mendasarkan inferensi tentang rate kegagalan dalam versi-versi terdahulu. Tentunya, setelah dideteksi setiap bug tidak akan dibiarkan berada dalam program, sehingga setiap versi akan diuji hingga tak ada lagi buug didalamnya. Selanjutnya, jadilah versi yang baru. Dengan demikian, suatu estimasi reliabilitas haruslah didasarkan pada pengujian-pengjian yang bebas dari kegagalan. Jika ingin diperoleh suatu tingkat kepercayaan bahwaw rate kegagalan kurang dari 10^{-9} kegagalan per jam, maka perlu menguji hingga lebih dari 10^9 jam tanpa adanya kegagalan.

Interval-interval kepercayaan untuk peluang kegagalan yang didasarkan pada pengujian yang bebas kesalahan dapat dilturunkan sebagai berikut :

Misalkan p menyatakan peluang yang tidak diketahui dari kegagalan pada suatu uji kasus yang diambil secara acak. Anggap bahwa n buah uji kasus telah dilakukan tanpa ditemukan suatu kegagalan. Umumnya, jika n besar dan p kecil, jumlah kegagalan akan merupakan suatu perubah acak X, dengan suatu distribusi Poison dengan rata-rata $\mu = np$:

$$P(X = x) = e^{-\mu} \mu^x / x! \quad ; x = 0,1,2, ... \qquad (4\text{-}1)$$

Jadi kita telah menyelidiki data dengan model distribusi yang mempunyai peluang

$$P(X = 0) = e^{-\mu} = e^{-np} \qquad (4\text{-}2)$$

Jika datanya tidak signifikan secara statistik hingga level α, maka nilai-nilai p harus memenuhi

$$\alpha <= P(X = 0) = e^{-np}$$

atau

$$p <= -\log \alpha / n \qquad (4\text{-}3)$$

yang menyatakan interval kepercayaan sebesar $100(1-\alpha)$ % untuk p jika n buah uji kasus dilakukan tanpa ada kegagalan.

Dalam Tabel 1 di bawah ini ditunjukan beberapa interval kepercayaan untuk tingkat-tingkat kepercayaan yang berbeda pula.

Tabel 1: Interval Kepercayaan

Tingkat Kepercayaan	Interval Kepercayaan
95 %	p < 3,00 / n
99 %	p < 4,61 / n
99,9 %	p < 6,91 / n
99,99 %	p < 9,21 / n
99,999 %	p < 11,5 / n

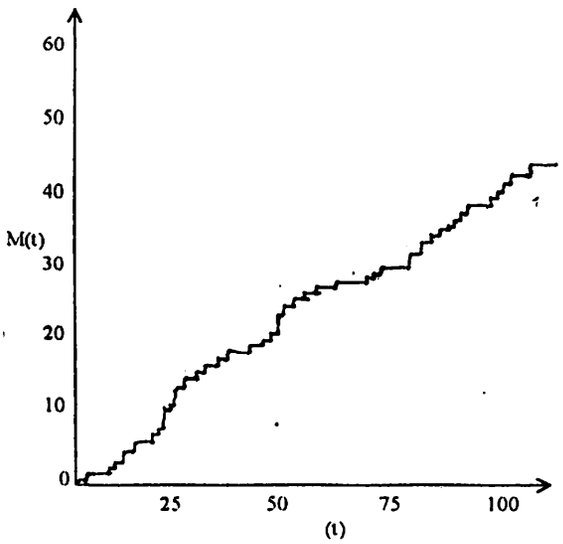
Sebagai contoh, untuk meyakini 99% bahwa peluang kegagalan adalah kurang dari 10^{-9} memerlukan $4,6 \times 10^9$ uji kasus tanpa kegagalan. Jika satuan waktunya dalam jam, maka hal ini sama dengan 525.000 tahun. Jadi, waktu pengujian yang sangat lama diperlukan untuk verifikasi reliabilitas yang tinggi. Tetapi, waktu pengujian yang sangat lama ini tentunya tidak mungkin dilakukan, karena untuk mengetahui distribusi input lapangan secara tepat adalah suatu masalah pula.

5. KESIMPULAN

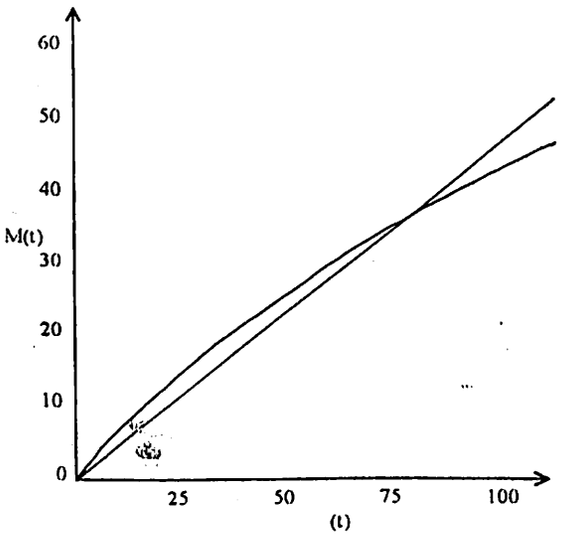
Adalah suatu hal yang sangat sulit untuk membuktikan sesuatu yang hampir tak mungkin, tetapi ada pertanda (kenyataan) bahwa suatu verifikasi statistik formal mengenai reliabilitas adalah dimungkinkan untuk keamanan beberapa sistem yang cukup kritis. Oleh karena itu, pengembangan perangkat lunak dan teknik-teknik evaluasi yang terbaik bisa terus dilakukan.

DAFTAR PUSTAKA

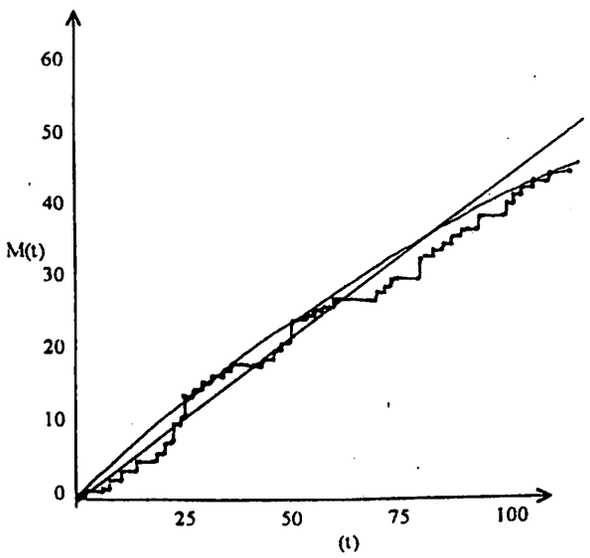
1. Currit P.A., M. Dyer, dan H.D. Mills : '*Certifying the Reliability of Software*', IEEE Transactions on Software Engineering, SE-12, 3-11, 1986.
2. Miller D.R. : '*Exponential Order Statistic Model of Software Reliability Growth*', IEEE Transactions on Software Engineering, SE-12, 12-24, 1986.
3. Musa J.D. dan K. Okumoto : '*A Logarithmic Poisson Execution Time Model for Software Reliability Measurement*', IEEE Computer Society Press, Washington D.C., 230-238, 1984.
4. Scholz F.W. : '*Software Reliability Modeling and Analisis*', IEEE Transactions on Software Engineering, SE-12, 25-31, 1986.



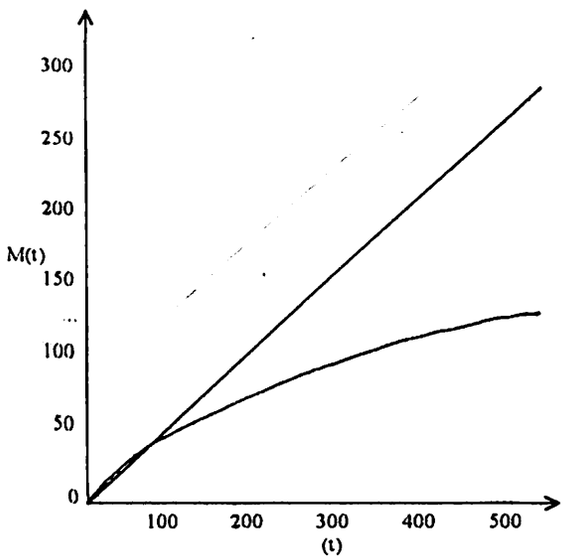
Gambar 1. Data Kegagalan Kumulatif



Gambar 2. Dua Model dengan Intensitas Monoton Lengkap



Gambar 3. Gabungan dari Model dan Data



Gambar 4. Ekstrapolasi Model