数理解析研究所講究録 1373 巻 2004 年 110-118

分散ソフトウェア開発環境下における最適リリース問題とその応用

鳥取環境大学・環境情報学部
情報システム学科
田村 慶信 (Yoshinobu Tamura)
Department of Information Systems, Faculty of Environmental and Information Studies, Tottori University of Environmental Studies
E-mail: tamura@kankyo-u.ac.jp

鳥取大学・工学部
社会開発システム工学科
山田 茂 (Shigeru Yamada)
Department of Social Systems Engineering, Faculty of Engineering, Tottori University
E-mail: yamada@sse.tottori-u.ac.jp

法政大学・工学部
経営工学科
木村 光宏 (Mitsuhiro Kimura)
Department of Industrial and Systems Engineering, Faculty of Engineering, Hosei University
E-mail: kim@k.hosei.ac.jp

概 要

近年の情報システムでは，インターネットの普及により世界中が同時に新しい情報を得ることができるようになり，実時間的あるいはインタラクティブ性の高い機能の追求へと関心が深まっているといえる．こうした状況から，ネットワークを基にしたソフトウェアの分散開発，およびソフトウェアそのものの分散化がさらに拡大してきた．

本論文では，こうした分散ソフトウェア開発環境を対象とした信頼性評価法および最適リリース問題について議論する．すなわち，分散開発環境において開発されたソフトウェアの信頼性評価例を示すとともに，総合テスト工程を終了して，実際の運用段階に移行するのに最適な時期を，総ソフトウェアコストを最小にするような総テスト時間を求める問題として議論する．さらに，分散開発環境を対象とした既存のソフトウェア信頼性評価ツールに最適リリース時刻を決定するための機能を付加する．本ツールは，オブジェクト指向型言語の 1 つである Java を用いて開発する．

1 まえがき

今日，世界的に IT 革命が進展し，あらゆる分野で構造的改革や組織改革が本格化する中で，ソフトウェアに依存する場面が多くなっており，ソフトウェア開発に大きな期待と関心が集まっている．しかし，現状のソフトウェア開発においては生産性や信頼性 (software reliability) に関する問題がさまざまな形で顕在化している．つまり，ソフトウェアの需要が急速に増加し，ソフトウェア開発がますます複雑になる一方で，その開発は人的作業によって開発されている．また，求められる品質が高くなる一方で，ソフトウェアのリリースサイクルが極端に短くなってきている．このような状況では，ソフトウェアの開発工程で多くのソフトウェアフォールト (software fault, 以下フォールトと略す) と呼ばれる欠陥や誤りが潜入することになる．このフォールトが潜在したソフトウェアは，利用者 (ユーザ) にとって，時に致命的な問題を引き起こす原因ともなる．このような現状の中で，ソフトウェアの代表的品質特性である信頼性の高いソフトウェアを開発することが重要な課題となっており，ソフトウェアシステムの高信頼化を図る問題において，ソフトウェアの信頼性を定量的に評価することは，開発工程を科学的に管理する上で特に重要である．

現在，ソフトウェア開発環境は，インターネットの普及により，ソースコードなどをネットワーク上の領域へ保存することも当り前のようになってきている．例えば，CVS (Concurrent Versioning System) のように，ソースコードのバージョン管理システムなども普及してきており，開発環境の分散化が進んでいる．このように，開発環境の分散化が進むと，開発者はどこにいても，ネットワークを利用してソースコードをダウンロードしたり，リモートマシンにログインして開発を行うことができる．したがって，スタンドアローンで開発していた頃と比較して，自由な環境で開発ができるようになり，プロジェクトチームを組んで開発を行う場合にも，ネットワークを介

した分散化がなされていれば共同作業が容易になる．さらに，ソフトウェアの分散開発における技術として注目されている，オブジェクト指向の概念が拡大しつつある．この技術による開発作業量の大幅削減や，ソフトウェアの品質および生産性改善への効果が極めて大きいことから，今後広域に分散化されたワークステーションにより，並行して開発されたオブジェクトを利用した分散開発が発展していくものと考えられる．ソフトウェアのあり方も，このようなソフトウェア開発環境の急激な変化に柔軟に対応した大規模な分散化ソフトウェアシステムの構成法，さらには，そのときの全体の品質の確保などが大きな問題となっている．品質，生産性に対する要求が一段と厳しくなっている中で，ネットワークで相互接続された分散開発環境のもとで開発されたソフトウェアシステムの信頼性評価とともに，開発期間の短縮および開発コストの削減が重要かつ現実的な課題となる [1, 2, 3].

本論文では，こうした分散ソフトウェア開発環境を対象とした一般化確率微分方程式モデルに基づく信頼性評価法および最適リリース問題について議論する．すなわち，分散開発環境において開発されたソフトウェアの信頼性評価例を示すとともに，総合テスト工程を終了して，実際の運用段階に移行するのに最適な時期を，総ソフトウェアコストを最小にするような総テスト時間を求める問題として議論する．特に，従来の最適リリース問題は，ソフトウェアコストの期待値を最小にする確定的な最適リリース時刻を求めるだけであった．しかし，本論文では，ソフトウェアコストを最小にするような最適リリース時刻の信頼区間を導出することにより，現実的な最適出荷時期を見積もることが可能となることを示す．さらに，分散開発環境を対象とした既存のソフトウェア信頼性評価ツールに最適リリース時刻を決定するための機能を付加する．本ツールは，オブジェクト指向型言語の1つであるJavaを用いて開発する [4]．また，JavaとMathematica[1]を完全かつ透過的に統合するJ/Link機能を用いる．

2 分散開発環境に対するソフトウェア信頼性モデル

分散開発環境下におけるソフトウェア開発の特徴として，オブジェクト指向技術によるソフトウェアの部品化が挙げられる．このソフトウェアの部品化は，ソフトウェアそのものを標準化して再利用することを可能とし，さらに，この技術による開発作業量の大幅削減やソフトウェアの品質および生産性改善への効果が極めて大きいものと期待されている [2]．分散開発環境のように対象とするソフトウェアシステムが大規模な場合や，中に含まれる記述プログラム言語の種類やシステム内部の構成要素数が多い場合，各ソフトウェアコンポーネント間の相互作用もより一層顕在化することから，テスト工程における発見フォールト数の挙動が把握しにくくなる [3]．従来から，ソフトウェア製品の開発プロセスにおける出荷品質の把握やテスト進捗管理のための信頼性評価を行うアプローチとして，ソフトウェア故障の発生現象を不確定事象としてとらえて確率・統計論的に取り扱う方法がとられてきた．その1つがソフトウェア信頼度成長モデル（software reliability growth model，以下SRGMと略す）である [5].

分散開発環境を対象とした，より現実的なフォールト発見過程を考慮した一般化ソフトウェア信頼度成長モデルとして，以下に示すような分散開発環境の総合テスト工程におけるフォールト発見過程を，計数過程として扱った非同次ポアソン過程（nonhomogeneous Poisson process，以下NHPPと略す）に基づく一般化SRGMと，そのフォールト発見過程を連続的に変動していく確率過程でモデル化した確率微分方程式（stochastic differential equation，以下SDEと略す）[6] に基づく一般化SRGMが提案されている．

(a) 分散開発環境に対するNHPPに基づいた一般化SRGM[7]（一般化NHPPモデルと略す）

$$H_{dde}(t) = a\left\{\sum_{i=1}^{n}\frac{p_i(1-e^{-b_i t})}{1+c_i\cdot e^{-b_i t}}\right\}$$
$$\left(a>0, b_i>0, p_i>0, \sum_i p_i = 1\ (i=1,2,\cdots,n)\right). \quad (1)$$

ここで，$H_{dde}(t)$はテスト時刻tまでに発見・除去された総フォールト数の期待値を表す．また，aはテスト開始前のソフトウェア内の潜在フォールト数の期待値，$b_i\,(i=1,2,\cdots,n)$はi番目のコンポーネントに対する残存フォールト1個当りのフォールト発見率，$p_i\,(i=1,2,\cdots,n)$はi番目のコンポーネントに対する重み，

[1] Mathematicaは米国Wolfram Research社の登録商標である．

つまりテストの重要度を表す．$c_i\,(i=1,2,\cdots,n)$ は $(1-l_i)/l_i$ により表されるもので，l_i をフォールト発見能力に関するテスト習熟係数という．

(b) 分散開発環境に対する SDE に基づいた一般化 SRGM[7] （一般化 SDE モデルと略す）

$$\mathrm{E}[N_{dde}(t)] = m_0\left[1-\left\{\sum_{i=1}^{n}\frac{p_i e^{-b_i t}(1+c_i)}{1+c_i e^{-b_i t}}\right\}e^{\frac{\sigma^2}{2}t}\right]. \tag{2}$$

ここで，$N_{dde}(t)$ はテスト時刻 t までに発見・修正された総フォールト数を表し，式 (2) はその期待値である．また，m_0 はテスト開始前に潜在していた総期待フォールト数を，$b_i(i=1,2,\cdots,n)$ は i 番目のコンポーネントに対する残存フォールト 1 個当りのフォールト発見率を，パラメータ $p_i(i=1,2,\cdots,n)$ は i 番目のコンポーネントに対する重み，すなわちテストにおける重要度を表している．$c_i\,(i=1,2,\cdots,n)$ は $(1-l_i)/l_i$ により表されるもので，l_i をフォールト発見能力に関するテスト習熟係数という．また，σ は未知の定数パラメータである．

式 (1) および式 (2) により記述される両者のモデルは分散開発環境を対象としたものであり，コンポーネントが n 個使われたものと仮定している．特に，本モデルは，過去に提案されたソフトウェア信頼度成長モデル [8, 9] に比べて，重みパラメータ $p_i(i=1,2,\cdots,n)$ の値に応じて，指数形成長曲線から S 字形成長曲線までの累積発見フォールト数に関する広い範囲の特性曲線を形成することができる．これにより，既存の SRGM より高い精度で信頼性評価尺度の推定が可能となるだけでなく，開発管理者にとっては，適用モデルの選定作業の省力化に結びつくことが確認されている [10].

本論文では，分散開発環境におけるソフトウェア信頼性を定量的に評価するために，より現実的なフォールト発見過程を考慮した式 (2) における一般化 SDE モデルに基づいた最適リリース問題について議論する．特に，テスト工程およびソフトウェアの運用段階に費やす総ソフトウェアコストの関数を確率変数として扱い，総ソフトウェアコストおよび最適リリース時刻の統計的信頼区間に基づく存在範囲を用いることにより，総ソフトウェアコストを最小にする最適リリース時刻の推定を数値例を用いて行う．

3 ソフトウェアの最適リリース問題

ソフトウェアの運用段階における品質は，実施されるテストの技法および総テスト時間に依存する．つまり，ソフトウェアの高信頼性を実現させるために，テスト時間を増加させれば，ソフトウェア内に潜在する多くのフォールトを発見・修正・除去することが可能であり，運用段階におけるソフトウェアの信頼性は向上する．しかし，長時間のテストを実施すれば，その分ソフトウェアの信頼性は向上するが，テストに費やすコストが増加する．逆に，テストの実施時間を減少させると，テストに費やすコストは減少し，ソフトウェア製品の出荷も早くなるが，ソフトウェア内の残存フォールト数が増加して運用段階での保守コストが増大することになる．よって，ソフトウェアをテスト工程から運用段階に移行させるのに適切な総テスト時間を決定するには，信頼性とコストのトレードオフの関係を考慮しなければならない．このようなユーザへの引渡し（出荷）の適切な時期を決定する問題を，ソフトウェアの最適リリース問題という．また，分散開発環境では，コンポーネントごとのテストを行う単体テスト工程と，単体テストの終了したコンポーネントを組み合わせてテストを行う段階，すなわち，各ソフトウェアコンポーネントを結合した後の最終段階に位置付けられる総合テスト工程との関係を考える必要もある．

一般に，運用段階におけるフォールト修正に伴う保守コストは，テスト工程における保守コストよりも大きい．したがって，ソフトウェアの最適リリース問題として，テスト工程および運用段階における保守コストを考慮して，総ソフトウェアコストが最小となるように最適リリース時刻を求めることが考えられる [11, 12].

本章では，総期待ソフトウェアコストを評価基準として，分散開発環境に対する一般化された確率微分方程式モデルに基づくコンポーネントの再利用率を考慮したソフトウェアの最適リリース問題について議論する

3.1 ソフトウェアコストの定式化

本節では，2 で議論した SDE モデルを用いることによって，分散開発されたソフトウェアの開発管理上の問題への応用の 1 つとして，テスト工程および運用段階でのコスト要因を挙げ，総ソフトウェアコストを評価基準として，最適な出荷時期を決定するための最適リリース問題について議論する．

総ソフトウェアコストを定式化するために，以下のコストパラメータを定義する．

c_{1i}: コンポーネント i の単体テストで発見されるフォールト 1 個当りの修正コスト $(c_{1i} > 0)$,

c_{2i}: コンポーネント i の単体テストでの単位時間当りのテストコスト $(c_{2i} > 0)$,

c_{1c}: 総合テストで発見されるフォールト 1 個当りの修正コスト $(c_{2c} > 0)$,

c_{2c}: 総合テストでの単位時間当りのテストコスト $(c_{1c} > 0)$,

c_{3c}: リリース後に発見されるフォールト 1 個当りの保守コスト $(c_{3c} > 0, c_{3c} > c_{1i}, c_{3c} > c_{2c})$.

ここでは，各サブシステムについて累積発見フォールト数データの成長曲線の形状により，以下に示す NHPP に基づく習熟 S 字形 SRGM を用いた信頼性評価法の適用を前提とする [5]:

$$H(t) = \frac{a(1-e^{-bt})}{1+c\cdot e^{-bt}} \quad (c>0). \tag{3}$$

ここで，a はテスト開始前に潜在していた総フォールト数，b は残存フォールト 1 個当りのフォールト発見率，c は $(1-l)/l$ により表されるもので，l をフォールト発見能力に関するテスト習熟係数という．

さらに，各コンポーネントの総合テスト工程への遅延の影響による予定どおりに運用できないために発生したユーザに対するペナルティ（補償）などを考える必要性があり，単体テスト終了時刻が総合テスト開始時刻を越えた場合にはペナルティコストが課せられるものとする．コンポーネント i $(i = 1, 2, \cdots, n)$ の，単体テストから総合テスト工程への引き渡し遅延時間 $(t_i - t_{di})$ に対するペナルティコスト関数を，以下のように定式化する．

$$G_i(t_i) = \begin{cases} c_{3i}\left\{e^{k_i(t_i - t_{di})} - 1\right\} & (t_i > t_{di}) \\ 0 & (t_i \le t_{di}) \end{cases}. \tag{4}$$

ここで，t_{di} はコンポーネント i の単体テスト開始時刻から総合テスト開始時刻までの時間 $(t_{di} > 0)$ を表し，$c_{3i}(> 0)$ および $k_i(> 0)$ は定数パラメータを表す．よって，以下のような各コンポーネントのテストに要する期待コストが得られる．

$$C_i(t_i) = c_{1i}H_i(t_i) + c_{2i}t_i + G_i(t_i) \qquad (i = 1, 2, \cdots, n). \tag{5}$$

ここで，$H_i(t_i)$ は i 番目のコンポーネントに対して適用された NHPP に基づく指数形 SRGM または遅延 S 字形 SRGM の平均値関数を表す．さらに，ペナルティコスト関数において，コンポーネント i の単体テスト終了時刻が総合テスト開始時刻を越えない場合を，

$$t_{di} = t_i,$$

と仮定する．

これにより，総ソフトウェアコストを

$$Cost(N_{dde}(t), t) = \sum_{i=1}^{n} C_i(t_i) + c_{2c}t + c_{1c}N_{dde}(t) + c_{3c}\{m_0 - N_{dde}(t)\}, \tag{6}$$

により定式化する．$N_{dde}(t)$ は確率変数であることから，$Cost(N_{dde}(t), t)$ は確率変数となる．式 (6) を変形して，

$$Cost(N_{dde}(t), t) = \sum_{i=1}^{n} C_i(t_i) + c_{2c}t - (c_{3c} - c_{1c})N_{dde}(t) + m_0c_{3c}, \tag{7}$$

とする．ここで，$N_{dde}(t)$ の分布関数が式 (2) の一般化 SDE モデルの場合は

$$\Pr[N_{dde}(t) \le n] = \Phi\left(\frac{\log\frac{m_0}{m_0-n} + \log\left[\sum_{i=1}^{n}\frac{p_ie^{-b_it}(1+c_i)}{1+c_ie^{-b_it}}\right]}{\sigma\sqrt{t}}\right), \tag{8}$$

となるので，この分布関数を用いて $Cost(N_{dde}(t),t)$ の分布関数を考える．式 (7) より，

$$N_{dde}(t)=\frac{\sum_{i=1}^{n}C_i(t_i)+c_{2c}t+m_0c_{3c}-Cost(N_{dde}(t),t)}{c_{3c}-c_{1c}}, \tag{9}$$

となる．式 (8) に式 (9) を代入して変形する．ここで，

$$C=-n(c_{3c}-c_{1c})+(c_{2c}t+m_0c_{3c}), \tag{10}$$

とおくと，$Cost(N_{dde}(t),t)$ の分布関数は，

$$\begin{aligned}&\Pr[Cost(N_{dde}(t),t)\le C]\\ &=\ 1-\Phi\left(\left\{\log\left[m_0(c_{3c}-c_{1c})\Big/\left\{C-\left(\sum_{i=1}^{n}C_i(t_i)+c_{2c}t+m_0c_{1c}\right)\right\}\right]\right.\\ &+\ \left.\log\left[\sum_{i=1}^{n}\frac{p_ie^{-b_it}(1+c_i)}{1+c_ie^{-b_it}}\right]\right\}\Big/\sigma\sqrt{t}\Bigg),\end{aligned} \tag{11}$$

となる．

さらに，総期待ソフトウェアコストは，式 (6) の期待値をとることにより以下のように表される．

$$\mathrm{E}[Cost(N_{dde}(t),t)]=\sum_{i=1}^{n}C_i(t_i)+c_{2c}t+c_{1c}\mathrm{E}[N_{dde}(t)]+c_{3c}(m_0-\mathrm{E}[N_{dde}(t)]). \tag{12}$$

3.2 ソフトウェアコストの信頼区間と最適リリース時刻

本論文では，ソフトウェアコストの関数を確率変数として扱っていることから，コストの分布関数の値が $(1-0.01\alpha)/2$ となるコストと，$(1+0.01\alpha)/2$ となるコストを求め，ソフトウェアコストの α% 区間を求めることが可能となる．この区間を，ソフトウェアコストの α% 信頼区間とする．このとき，ソフトウェアコストの α% 信頼区間の上下限は，それぞれ

$$C_U(t) = \mathrm{E}[Cost(N_{dde}(t),t)]+\beta_1(t)\sqrt{\mathrm{Var}[Cost(N_{dde}(t),t)]}, \tag{13}$$

$$C_L(t) = \mathrm{E}[Cost(N_{dde}(t),t)]-\beta_2(t)\sqrt{\mathrm{Var}[Cost(N_{dde}(t),t)]}, \tag{14}$$

と与えられる．ここで，$C_U(t)$ および $C_L(t)$ は，それぞれソフトウェアコストの α% 信頼区間の上限および下限である．但し，$\beta_1(t)$, $\beta_2(t)$ は，それぞれ時刻 t における標準正規分布関数の $100(1\pm\alpha)/2$ パーセント点である．

次に，最適リリース時刻について議論する．最適リリース時刻を T^* とすると，T^* は式 (12) の総期待ソフトウェアコストを最小にする時刻 $t=T^*$ を求めることで得られる．さらに，本論文では α% 信頼区間の $C_U(t)$ と $C_L(t)$ を最小にする時刻 $t=T_U^*$ および $t=T_L^*$ を求めることで，α% 信頼区間における最適リリース時刻の範囲を求めることができる．ここで，T_U^* は $C_U(t)$ を最小にする時刻，T_L^* は $C_L(t)$ を最小にする時刻である．

4 数値例

実際のテスト工程（総合テスト）において観測されたデータを適用した数値例を示す．ここに示す数値例は，実際にある企業で開発されたプロジェクトデータに基づいている．本論文で用いたデータは，総開発規模：約 271.4 Ksteps，新規開発・既存部改造：約 40.4 Ksteps，流用率：約 85%であり，9 つのソフトウェアコンポーネントから構成されたソフトウェアシステムのテスト工程から採取されたものである．また，テスト時間 t_k の測定単位は日である．

4.1 信頼性評価例

ここで，上記のデータに基づくモデルパラメータの推定結果は以下のようになる．

$$\hat{m}_0=42.573,\ \hat{b}_i=0.22526,\ \hat{b}_j=0.093545,\ \hat{\sigma}=0.047806.$$

本論文では，l_i をソフトウェアコンポーネントの再利用率として近似的に表現できるものと考え，$l_i = 0.85, l_{n+j} = 0.15$ とした．また，重みパラメータ $p_i (i = 1, 2, \cdots, n)$ については，最尤推定値を最大とする $p_i = 0.3, p_{n+j} = 0.7$ の値を採用した．このときの推定された残存フォールト数のサンプルパス $\widehat{M_{dde}}(t) (= \hat{m}_0 - \widehat{N_{dde}}(t))$ を図 1 に示す．

ソフトウェアの信頼性評価尺度として，残存フォールト数の分散および変動係数は重要である．その分散と変動係数は，それぞれ次のように導出できる．

$$\begin{aligned} \mathrm{Var}[M_{dde}(t)] &= \mathrm{Var}[N_{dde}(t)] \\ &= m_0^2 \left\{ \sum_{i=1}^{n} \frac{p_i e^{-b_i t}(1+c_i)}{1+c_i e^{-b_i t}} \right\}^2 e^{\sigma^2 t}(e^{\sigma^2 t} - 1), \end{aligned} \tag{15}$$

$$\begin{aligned} CV(t) &\equiv \frac{\sqrt{\mathrm{Var}[N_{dde}(t)]}}{\mathrm{E}[N_{dde}(t)]} \\ &= \frac{m_0 \left\{ \sum_{i=1}^{n} \frac{p_i e^{-b_i t}(1+c_i)}{1+c_i e^{-b_i t}} \right\}^2 e^{\sigma^2 t}(e^{\sigma^2 t} - 1)}{1 - \left\{ \sum_{i=1}^{n} \frac{p_i e^{-b_i t}(1+c_i)}{1+c_i e^{-b_i t}} \right\} e^{\frac{\sigma^2}{2} t}}. \end{aligned} \tag{16}$$

また，ソフトウェア故障の発生頻度の特性を調べるのに有用な平均ソフトウェア故障発生時間間隔（mean time between software failures, 以下 MTBF と略す）の尺度としての瞬間 MTBF（Instantaneous MTBF），およびテスト開始時点から考えたときの累積 MTBF（Cumulative MTBF）は，

$$MTBF_I(t) = \frac{1}{\mathrm{E}\left[\frac{dN_{dde}(t)}{dt}\right]}, \tag{17}$$

$$MTBF_C(t) = \frac{t}{\mathrm{E}[N_{dde}(t)]}, \tag{18}$$

として近似的に表される [9]．式 (17) および式 (18) における推定された MTBF を図 4 に示す．

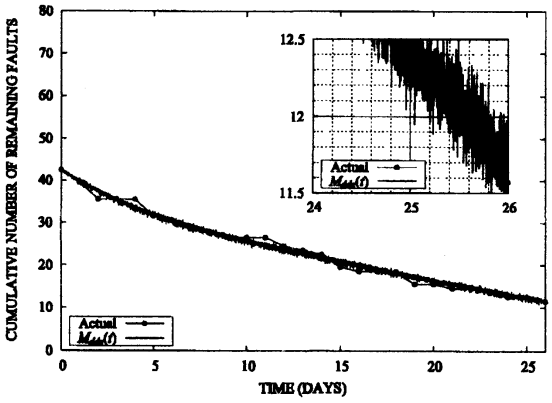

図 1： 推定された残存フォールト数のサンプルパス．

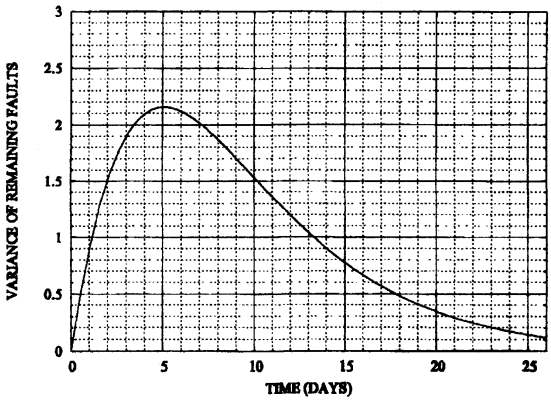

図 2： 推定された残存フォールト数の分散．

4.2 最適リリース問題

3.1 の総ソフトウェアコストを用いて，ソフトウェアの最適リリース問題の数値例を示す．ここでは 1 例として，以下のパラメータを設定して最適リリース時刻を求める．

$$c_{11} = 1,\ c_{12} = 1,\ c_{13} = 1,\ c_{14} = 1,\ c_{15} = 1,\ c_{16} = 1,\ c_{17} = 2,\ c_{18} = 1,\ c_{19} = 2,$$
$$c_{21} = 2,\ c_{22} = 2,\ c_{23} = 2,\ c_{24} = 2,\ c_{25} = 2,\ c_{26} = 2,\ c_{27} = 4,\ c_{28} = 2,\ c_{29} = 4,$$
$$c_{1c} = 10,\ c_{2c} = 20,\ c_{3c} = 50.$$

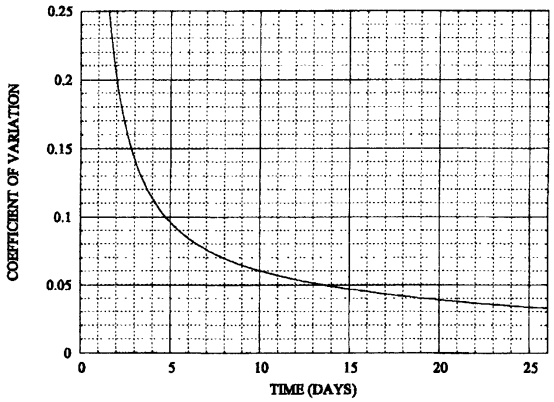

図 3： 推定された変動係数.

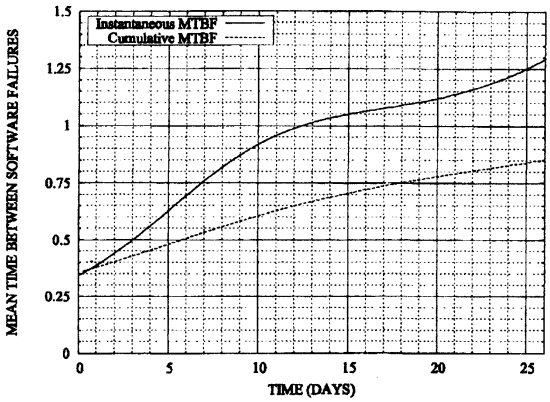

図 4： 推定された瞬間 MTBF および累積 MTBF.

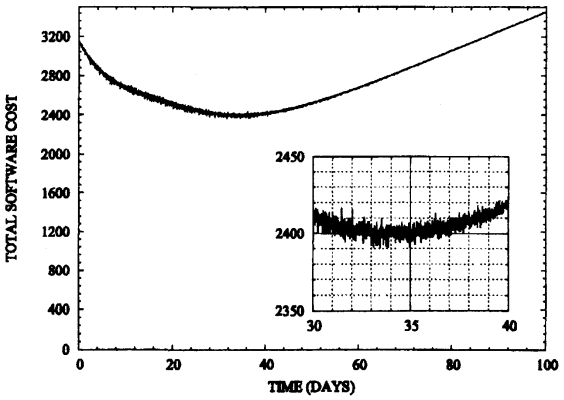

図 5： 推定されたソフトウェアコストのサンプルパス.

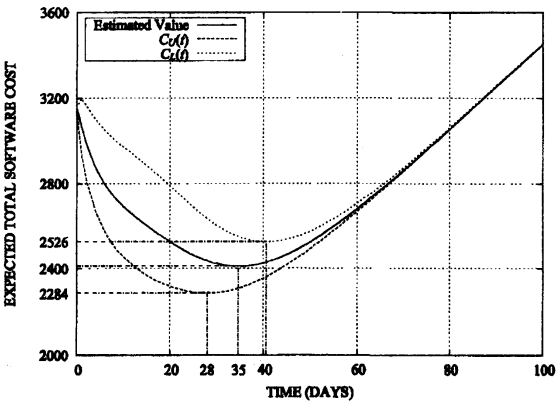

図 6： 推定された総期待ソフトウェアコスト.

このとき，4.1 の信頼性解析結果に基づいて推定されたソフトウェアコストのサンプルパスを図 5 に示す.

次に，推定された総期待ソフトウェアコストの時間変化の様子を図 6 に示す．図 6 から，最適リリース時刻は $T^* = 34.671$ と推定され，このときの総期待ソフトウェアコストは 2410.6 となった．また，ソフトウェアコストの関数を確率変数として扱っているので，ソフトウェアコストの分布関数の値が 0.05 となるコストと，0.95 となるコストを求めることにより，ソフトウェアコストの 90% 信頼区間を求めることが可能となる．図 6 には，ソフトウェアコストの 90% 信頼区間の変化が示されており，$\widehat{C_U}(t)$ および $\widehat{C_L}(t)$ においてソフトウェアコストが最小となる時刻はそれぞれ $T_U^* = 40.652$ および $T_L^* = 28.070$ となることが確認できる．したがって，90% 信頼区間におけるソフトウェアコストの存在範囲はそれぞれ $C_U(T_U^*) = 2525.6$ および $C_L(T_L^*) = 2283.7$ となることが分かる．開発管理者は，この情報をもとに，テスト工程を終了してユーザにソフトウェアを引き渡すのに最適な時期を定量的に把握することが可能となる．特に，統計的信頼区間に基づく存在範囲を用いることにより，より現実的な最適リリース時刻およびソフトウェアコストの見積りが可能となる.

5 ソフトウェア信頼性評価システムの開発

本論文で開発するツールは，オブジェクト指向型言語の 1 つである Java に加え，数値計算の過程に対して，Mathematica による J/Link 機能が使用されている．世界で唯一完全に統合された技術演算システムである Mathematica は，四則演算，関数の計算，ベクトル・行列の計算をはじめ微分・積分，n 次方程式の科学技術計算を行うことが可能な解析用ソフトであり，今日，産業界，行政機関，および教育界の世界中の 100 万人以上のユーザに使用されている．特に，最近の新しい機能として拡充された Mathematica と Java を完全かつ透過的に統合する

J/Link 機能は，Java アプリケーションの全ての機能を Mathematica から使用し，同時に Java のプログラムから Mathematica の機能を使用することができる．したがって，Mathematica を用いた研究成果を，ソフトウェア開発ツールとして短期間でソフトウェア開発管理者に提供することが可能となる．

本ツールの追加機能における実行手順を以下に示す．

step *1.* 分散開発環境の総合テスト工程から採取された累積発見フォールト数に関するデータファイルを読み込む．

step *2.* 既存の NHPP モデルおよび SDE モデル，**2** で議論した一般化 NHPP モデルおよび一般化 SDE モデルに対して，リンクされた Mathematica Kernel から最適リリース時刻および総期待ソフトウェアコストの推定を行う．

step *3.* 最適リリース問題のコスト評価基準として，NHPP モデルおよび一般化 NHPP モデルに対しては，総期待ソフトウェアコストおよびソフトウェア信頼度の推定結果をグラフ表示し，SDE モデルおよび一般化 SDE モデルに対しては，総期待ソフトウェアコストおよび変動係数の推定結果をグラフ表示する．

本論文では，既存のソフトウェア信頼性評価ツールに対して最適リリース時刻を決定するための機能を付け加える．これは，観測データの読み込み，モデルの選択，最適リリース時刻の推定および総期待ソフトウェアコストの推定，および観測データと推定結果のグラフ表示といった手順で進行する．開発されたソフトウェアツールの実行画面を図 7 に示す．

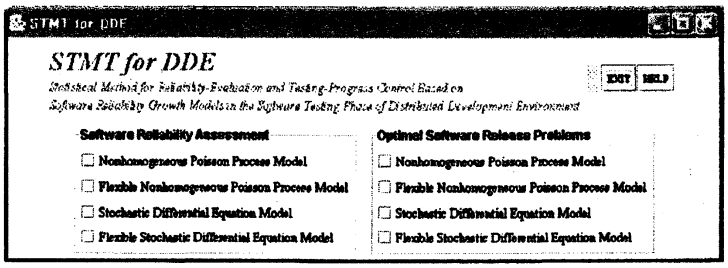

図 7： 本研究におけるツールの実行画面．

6 むすび

本論文では，Itô 型確率微分方程式を導入した分散開発環境に対する一般化 SRGM から，テスト工程で発見されたソフトウェアフォールト数に関するデータ解析を行って信頼性評価を実施した後に，運用段階へ移行するのに最適な時期を決定するという分散開発環境を対象としたソフトウェアの最適リリース問題について議論した．本論文では，総期待ソフトウェアコストを最小にする時刻を求めるだけでなく，最適リリース時刻を決定するための評価基準であるソフトウェアコストを確率変数として扱うことにより，ソフトウェアコストの α% 信頼区間を求めることができた．この α% 信頼区間を用いることにより，最適リリース時刻および総期待ソフトウェアコストの存在範囲を確認し，ソフトウェア開発管理上において重要となるより現実的な最適リリース時刻を求めることが可能となる．さらに，各ソフトウェアコンポーネントの総合テスト工程に対する納期遅れのペナルティコストを考慮したことにより，最適リリース問題の解として得られるより現実的な最適出荷時期を見積もることが可能となり，それを規定の納期と比較することによって開発管理者の意思決定の一助にすることができる．

本論文では，ソフトウェアコストの関数として，基本的な要因だけを取り上げて議論したが，ソフトウェア信頼性評価尺度といった別の要因を取り上げて評価することが今後の課題として挙げられる．

謝辞

本研究の一部は，文部科学省科学研究費基盤研究 (C)(2)（課題番号 15510129）の援助を受けたことを付記する．

参考文献

[1] A. Umar, *Distributed Computing and Client-Server Systems*, Prentice Hall, New Jersey, 1993.

[2] 高橋 宗雄, クライアント/サーバシステム開発の工数見積り技法～工数見積りモデルの適用法～, ソフト・リサーチ・センター, 東京, 1998.

[3] 赤羽豊和, クライアント/サーバ・システムのテスト技法, ソフト・リサーチ・センター, 東京, 1998.

[4] S. Holzner, *Java Programming: Black Book*, Impress, Tokyo, 2000.

[5] 山田　茂, ソフトウェア信頼性モデル－基礎と応用－, 日科技連出版社, 東京, 1994.

[6] L. Arnold, *Stochastic Differential Equations–Theory and Applications*, John Wiley & Sons, New York, 1974.

[7] 田村慶信, 内田雅也, 山田茂, 木村光宏, "分散開発環境に対する確率微分方程式に基づくソフトウェア信頼度成長モデルの一般化," 信頼性・品質3学会合同シンポジウム発表報文集, pp. 113-118, 2002年11月.

[8] M. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996.

[9] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software reliability measurement and assessment with stochastic differential equations," *IEICE Trans. Fundamentals*, vol. E77-A, no. 1, pp. 109-116, Jan. 1994.

[10] M. Uchida, Y. Tamura, and S. Yamada, "Software Reliability Analysis and Optimal Release Problem Based on a Flexible Stochastic Differential Equation Model in Distributed Development Environment," Proceedings of the 8th ISSAT International Conference on Reliability and Quality in Design, Honolulu, Hawaii, U.S.A., pp. 12-16, August 7-9, 2003.

[11] S. Yamada and S. Osaki, "Cost-reliability optimal release policies for a software system," *IEEE Trans. Reliability*, vol. R-34, no. 5, pp. 422-424, Dec. 1985.

[12] S. Yamada and S. Osaki, "Optimal software release policies with simultaneous cost and reliability requirements," *European J. Operational Research*, vol. 31, no. 1, pp. 46-51, July 1987.