ソフトウェアのインクリメンタル開発における信頼性評価モデルに関する研究

椎名, 紘伸 / SHIINA, Hironobu

(発行年 / Year)
2014-03-24
(学位授与年月日 / Date of Granted)
2014-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

2013 年度修士論文

ソフトウェアのインクリメンタル開発における信頼性評価モデルに関する研究

法政大学工学研究科
システム工学専攻

12R6206
椎名紘伸
Hironobu SHIINA

指導教員　木村　光宏 教授

概要

本研究では, 最も基本的で一般的なソフトウェアの開発方式として知られる，ウォーターフォールモデルに即したソフトウェア開発のテスト工程において, 従来からそのソフトウェアの信頼性評価モデルとして用いられてきたソフトウェア信頼性成長モデルを発展させた, ソフトウェアのインクリメンタル開発に適したソフトウェア信頼性成長モデルを提案し考察する．具体的には，既存の論文内で使われたソフトウェアメトリクスデータを用いて，最小二乗法よって構築されたモデルと最尤法よって構築されたモデルを複数提案した上で, 各モデルの有用性を評価する．

Abstract

本研究では, 最も基本的で一般的なソフトウェアの開発方式として知られる，ウォーターフォールモデルに即したソフトウェア開発のテスト工程において, 従来からそのソフトウェアの信頼性評価モデルとして用いられてきたソフトウェア信頼性成長モデルを発展させた, ソフトウェアのインクリメンタル開発に適したソフトウェア信頼性成長モデルを提案し考察する. 具体的には，既存の論文内で使われたソフトウェアメトリクスデータを用いて，最小二乗法よって構築されたモデルと最尤法よって構築されたモデルを複数提案した上で,各モデルの有用性を評価する.

目次

表目次

図目次

1　はじめに

1.1　研究の背景

ソフトウェア内に潜在するフォールト数を高い精度で推定したいという目的のため, 従来から数多くのソフトウェア信頼性モデルが開発されてきた. 中でも, ソフトウェア開発の最終段階であるテスト工程において採取される, ソフトウェアテストの時間と累積発見フォールト数のデータを用いて, 時系列解析的にソフトウェア内の累積フォールト数などの信頼性を評価するモデル, つまりソフトウェア信頼性成長モデルは近年, よく知られたものとなりつつある. 一方, ソフトウェア開発手法, 特にソフトウェア開発プロセスの改善も勢力的に行われている. それらは古典的なウォーターフォール型の開発プロセスを脱却し, 開発途中での仕様の変更など, ウォーターフォール型開発プロセスでは対応が困難である要求に応えようとしているものである. それらの中でアジャイル (agile ; 機敏な) 開発手法は近年注目されており, 比較的小規模なソフトウェアの開発に用いられてきている. 本研究では, アジャイル開発の 1 つとして知られる, インクリメンタル開発に着目し, そこから得られるデータに基づいて, ソフトウェアの信頼性を評価するモデルについて検討する. 特に, このプロセスは, ウォーターフォール型プロセスに比べて, 長期に渡る時系列データが得られないという特徴がある. これは時系列が短いデータが数本得られるというものになっており, 従来のソフトウェア信頼度成長モデルがうまく適用できない場合もありうる. 本研究ではこの点を回避し, また, ソフトウェア開発の上流工程から別途得られる情報を組み入れる手法についても検討する [1].

2　ソフトウェア開発プロセス

2.1　ウォーターフォールモデル

ウォーターフォールモデルはソフトウェア開発プロセスの 1 つで、最も基本的で一般的な開発である. これはソフトウェアの開発過程をいくつかの工程に分解して，各工程の終了時には文書を作成して次の工程に移るというものである．滝の水が流れ落ちる様子にたとえ, ウォーターフォールモデルと呼ばれる．これは要件定義，外部設計，内部設計，プログラミング，テストの 5 つの工程からなる．ウォーターフォールモデルは滝の水が逆流することがないのと同じように，次の工程に進んだら前のステップに逆戻りをしないように開発を進める．これにより，全体を見通すことができ，スケジュールの立案や資源配分，進捗状況の理解が容易にできるなどの点から, 高い信頼性が要求される大規模なシステムを開発する際に, 有効な代表的手法として使われ続けている．ソフトウェア開発の初期において要求定義を明確にできる状況において適切な開発モデルである.

2.1.1　ウォーターフォールモデルの工程

以下に文献 [2] の記述に基づいて, ウォータフォールモデルの各工程について説明する.

1. **要件定義** (開発するシステムの目的など, システムの計画を設計する.)
   経営課題や業務課題を通じて，情報システムの目的とシステムの目的と達成目標を明らかにする．業務の分析を通じて，適切な業務のあり方を考えながら，開発するシステムの機能を中心に，要求される様々な要求事項を決定する．最終的には，これらの要求事項と，システムの実現可能性，システム構成案，開発スケジュールなどのシステムの計画を合わせて，要求定義書にまとめる.

2. **外部設計** (ユーザから見たシステムの設計を行う.)
   新しいシステムを使った場合に業務がどのように変わるかを考えながら, システムの画面や帳票，周辺の情報システムとの間のインタフェースなどの設計を行う．システムで使うコードの設計や, ファイルやデータベースの設計を行うのも, この工程であり, 外部報告書を作成する.

3. **内部設計** (開発を進めるうえでのシステムの内部構造・仕組みを設計する.)
   外部設計に対して，内部設計は外部設計で決めた機能を実現するための「仕組み」をプログラミングの観点から設計する．また, 開発されるプログラムに対するテストの方法やスゲジュールなど, テストの計画を行う.

4. **プログラミング** (プログラム設計を行い, 実際にコーティングする.)
   初めに, コーティングを実施する際の標準ルールを確立する. 同じルールを用いてコーディングを行うことによって，スムーズに開発を進めることができるようにするためである．次に, 内部設計で決められたプログラムの詳細機能の使用に基づいて個々のプログラム単位であるモジュールの仕様を作成し, 各モジュールをコーディングしてモジュールを作成していく．一般に, この工程が最も人手のかかる工程となる.

5. **テスト** (テストを実施してソフトウェアの品質を高める.)
   単体テスト，結合テストに加え，システム全体を統合するシステムテスト, 新システムの下での業務遂

行を確認する運用テストなどがある．外部設計や要求定義書に記載された機能や要件が期待通りに機能し, 新システムのもとで業務をスムーズに行うことが出来るかどうかをユーザを交えて検証する．

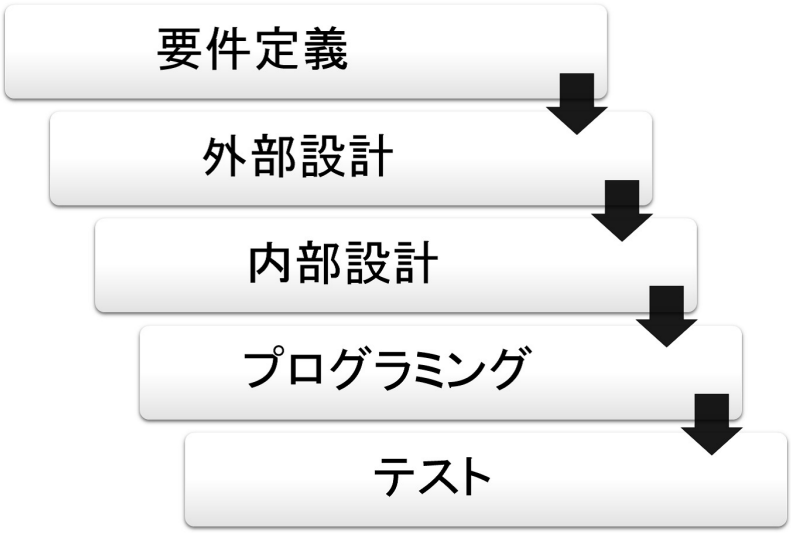

図 1　ウォーターフォールモデル．

2.2　インクリメンタル開発

インクリメンタル開発とは, システム全体を部分に分割してそれらを段階的に開発し, コアとなる部分に対して機能拡張を行う形で全体に統合していく開発方法である．分割した部分のそれぞれをインクリメントという．ウォーターフォールモデルでは, ソフトウェアの開発工程を工程ごとに明確に区別している．そのため, 開発の進捗管理がしやすく大規模システムの開発においても, 信頼性の高いシステムを構築できるメリットがある．しかしながら, 一方でデメリットもある．ウォーターフォールモデルは, コーティングを始めるまでにすべての要求と設計が確定していることを求めている．しかし, 要求事項や詳細の設計をすべて明確にすることは簡単ではなくまた, システムへの要求事項も時々刻々と変化しているのが普通である．ウォーターフォールモデルの欠点を克服するため, インクリメンタル開発が提案され, 実践されている [2].

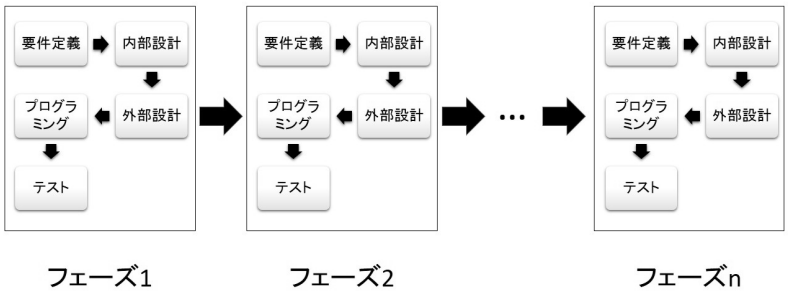

図 2　インクリメンタル開発.

2.2.1　インクリメンタル開発の例

以下に, インクリメンタル開発の例について説明する [2].

1. スパイラルモデル
   ソフトウェア開発プロジェクトを分割して, ウォーターフォールプロセスを何度も回しながら, 評価を繰り返し, ソフトウェアを成長させるモデルである. スパイラルという言葉の通り, ひと巻きごとにウォーターフォールのプロセスを経て, ソフトウェアを成長させていく.

2. 発展的プロトタイピングモデル
   もっとも重要な部分からシステムのプロトタイプを作成し, ユーザの要求を取り入れながら改良を続け, その繰り返しによってソフトウェアを成長させるモデルである. 要求事項を最初から定義し, 理解することが困難であることを想定しており, 開発工程を進めるにしたがってシステムのコンセプトを明らかにしていく. 未知の業務分野でのシステム開発など要求事項の理解が困難な場合に有効である.

2.3　先行研究

本研究ではインクリメンタル開発の予測モデルを提案する. 先行研究 [1] では, 観測されたすべてのメトリクスデータを使用し信頼性を評価するモデルを構築している. これに対して本研究ではすべてのメトリクスデータを使用するのではなく, フォールト数の期待値やフォールト発見率と相関の大きいメトリクスデータを選択し使用する. また, 先行研究ではフォールト数の期待値は例えば, 第 i フェーズにおいて第 $i-1$ フェーズまでのメトリクスデータを使用し推定し, フォールト発見率については, 現在のメトリクスデータを使用し各フェーズの推定を行った. 本研究では, フォールト数の期待値やフォールト発見率ともに, 最終フェーズである第 I フェーズの推定を第 1 フェーズから第 $i-1$ フェーズまでのメトリクスデータを使用し推定する. なぜなら, 実際のインクリメンタル開発においてもっとも重要なものは最終フェーズの予想データであるからである.

2.4 本論文の構成

本研究が提案する信頼性評価モデル構築に際して, 本論文の構成の概要について述べる．2 章ではソフトウェア開発プロセスで一般的に用いられるウォーターフォールモデルと本論文で用いるインクリメンタル開発の概要について述べる. 3 章で本論文で用いる評価モデルの構築方法およびを概要を示す. そして，文献 [1] で扱った，上流工程から得られるメトリクスデータの詳細について 4 章で述べる. 5 章では, インクリメンタル開発において適用可能なモデルを提案した上で, メトリクスデータを実際に使って，最小二乗法および最尤法によるモデルに含まれるパラメータの推定を行い, 信頼性評価尺度に関する計算例を示す. 6 章にて，本研究の結果および今後の課題について検討する.

3　モデリング

本研究では, NHPP モデルが当てはまると仮定し, 適用を試みる. また, 最小二乗法, 最尤法によりパラメータを推定しソフトウェア信頼性モデルを構築する. さらに, NHPP モデルから導出される信頼性評価尺度の 1 つである, MTBF を用いてソフトウェア信頼性モデルの評価をする. 以下では, 本研究で用いる NHPP モデル, 最尤法, MTBF についてそれぞれ説明する.

3.1　NHPP モデル [3, 4]

一般に, 実施されたテスト時間と発見された総フォールト数, または発生した総ソフトウェア故障数の関係は, 信頼度成長曲線 (reliability growth curve) によって把握され, ソフトウェア信頼性の改善や向上の変化・傾向を知ることができる. 指数関数的な傾向を示す指数形信頼度成長曲線, および S 字形関数的な S 字形信頼度成長曲線が典型的である. そこで, 指数形および S 字形信頼度成長曲線を示すフォールト発見事象やソフトウェア故障発生現象を記述するソフトウェア信頼度成長モデルは, それぞれ指数形ソフトウェア信頼度成長モデル (exponential software reliability growth curve) および S 字形ソフトウェア信頼度成長モデル (S-shaped reliability growth curve) と呼ばれ, 図 3 のように示される.

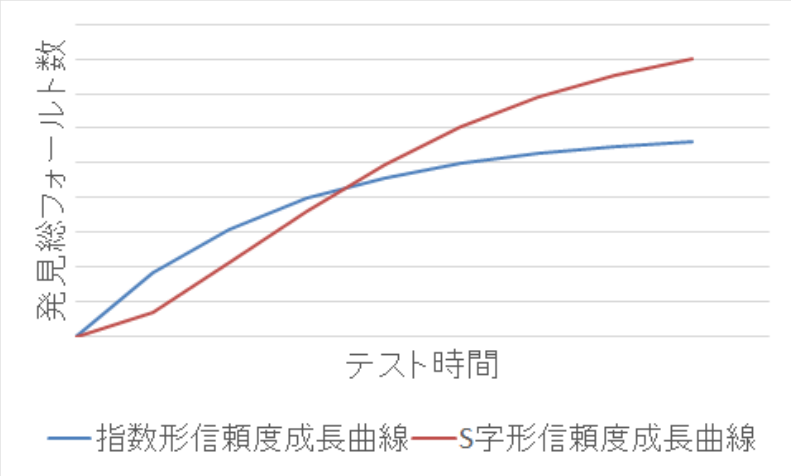

図 3　ソフトウェアの信頼度成長曲線.

このとき, 所定の時間区間内に発見されるフォールト数や発生するソフトウェアフォールト数や発生するソフトウェア故障数を観測して, これらの個数を数え上げる確率変数 $N(t)$ を導入し,

$$\Pr\{N(t)=n\}=\frac{\{H(t)\}^n}{n!}e^{-H(t)}\quad(n=0,1,\ldots), \tag{3.1.1}$$

で与えられる確率変数すなわちポアソン過程を仮定するソフトウェア成長モデルが NHPP モデルである. ポアソン過程の中でも, 次の 4 つの性質を満たすとき, $N(t)$ は非同次ポアソン過程あるいは非斉次ポアソン過程に従うとされる.

1. $N(0)=0$
   テスト時刻 $t=0$ ではフォールトは発見されない.

2. $\{N(t), \geq 0\}$ は独立増分をもつ
   相違なるテスト時間区間で発見されるフォールト数は統計的に独立である.

3. $\Pr\{N(t+\Delta t) - N(t) \geq 2\} = o(\Delta t)$ は独立増分をもつ
   任意の微小テスト時間区間 Δt で 2 個以上のフォールト数が発見される確率は無視できるほど小さい.

4. $\Pr\{N(t+\Delta t) - N(t) = 1\} = h(t)\Delta t + o(\Delta t)$
   任意の微小テスト時間区間 Δt で 1 個のフォールトが発見される確率は, テスト時刻 t におけるフォールト発見率またはソフトウェア故障の発生率 $h(t)$ に比例する.

上記の関数 $o(\Delta t)$ は微小時間 (Δt) の 2 次以上の高次の影響は無視できることを表し, $\lim_{\Delta t \to 0} o(\Delta t)/\Delta t = 0$ である. 性質 1〜4 を使い, 確率変数 $N(t)$ の確率分布を求めると式 (3.1.1) と同一の

$$\Pr\{N(t) = n\} = \frac{\{H(t)\}^n}{n!} e^{-H(t)}, \tag{3.1.2}$$

$$H(t) = \int_0^t h(x)dx, \tag{3.1.3}$$

を得る. ここで $H(t)$ は NHPP の平均値関数と呼ばれ, $N(t)$ の平均値すなわち時間区間 $(0, t]$ において発見される総期待フォールト数あるいは発生する総期待ソフトウェア故障数を表す. また, $h(t)$ は NHPP の強度関数 (intensity function) と呼ばれ, テスト時刻 t における瞬間フォールト発見率あるいはソフトウェア故障率を表す. 通常, 強度関数が時刻 t に関係なく一定, すなわち

$$h(t) = \lambda, H(t) = \lambda t \quad (\lambda > 0), \tag{3.1.4}$$

の場合は NHPP とは区別され, 確率関数 $N(t)$ は同次ポアソン過程 (homogeneous Poisson process) に従う. これに対して, もし, $H(t)$ が t の増加について有限な極限値を持つならば, たとえば, テストにより最終的に発見される総期待フォールト数, またはテスト開始時にソフトウェア内に潜在する総期待フォールト数を一定の $a(a > 0)$ とすれば

$$H(\infty) = \lim_{t \to \infty} H(t) = a, \tag{3.1.5}$$

であるので, 式 (3.1.2) 式 (3.1.3) より $N(\infty)$ の確率分布すなわち $N(t)$ の極限分布は

$$\lim_{t \to \infty} \Pr\{N(t) = n\} = \frac{a^n}{n!} e^{-a} \quad (n = 0, 1, 2, \ldots), \tag{3.1.6}$$

となる. 式 (3.1.6) は平均値が a のポアソン分布であることを意味し, NHPP はテストの経過時間に伴うソフトウェア内の残存フォールト数の確率的変動を考慮している. 式 (3.1.2) 式 (3.1.3) で定量化される NHPP モデルから, 信頼性評価に有用な定量的尺度を導出できる. まず, 任意のテスト時刻 t におけるソフトウェア内の残存フォールト数を $\bar{N}(t)$ により表すと

$$\bar{N}(t) = N(\infty) - N(t), \tag{3.1.7}$$

であるので, $\bar{N}(t)$ の平均値 $n(t)$ は式 (3.1.2) から

$$n(t) = E[\bar{N}(t)] = a - H(t), \tag{3.1.8}$$

となる,. また, 式 (3.1.8) は $\bar{N}(t)$ の分散 $v(t)$ にも一致することが示され

$$v(t) = \mathrm{Var}[\bar{N}(t)] = a - H(t), \tag{3.1.9}$$

である.

3.2 パラメータ推定

ソフトウェアの開発プロセスあるいは運用段階において計測・収集されたメトリクスデータに, ソフトウェア信頼性モデルを適用して信頼性評価を行うためには, いくつかの手順を踏まなければならないが, まずそのデータを解析してモデルパラメータを推定しなければならない. ソフトウェア開発工程では, ソフトウェア故障発生時間およびソフトウェアフォールト発見数データが観測される. ここで, 後者の形式のデータを使って, ソフトウェア信頼度成長モデルに含まれるパラメータの推定方法について述べる. また推定方法として, 最尤法 (method of maximum-likelihood) と最小二乗法 (method of least-squares) を取り上げる. 以下では最尤法について述べておく.

3.2.1 最尤法 [3]

一定のテスト時間間隔 $(0, t_k]$ において発見された総フォールト数 y_k に関する n 組の観測データ $(t_k, y_k)(k = 1, 2, \ldots, n)$ が観測されたものとする. このとき, このデータに対する平均値関数 $H(t)$ をもつ NHPP モデルの尤度関数は, NHPP の性質を用いて

$$\begin{aligned} L &= \Pr\{N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n\} \\ &= \exp\left[-H(t_n)\right] \Pi_{k=1}^{n} \frac{\{H(t_k) - H(t_{k-1})\}^{(y_k - y_{k-1})}}{(y_k - y_{k-1})!}, \end{aligned} \tag{3.2.10}$$

により与えられる. ここで, $t_0 = 0$, $y_0 = 0$ とする. 対数尤度関数は

$$\ln(L) = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[H(t_k) - H(t_{k-1})] - H(t_n) - \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!], \tag{3.2.11}$$

となる. 例えば

$$H(t) = m(t) = a(1 - e^{-bt}), \tag{3.2.12}$$

の平均値関数を持つ指数形ソフトウェア信頼度成長モデルの場合, 式 (3.2.11) の対数尤度関数は,

$$\ln(L) = y_n \ln a + \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[e^{-bt_{k-1}} - e^{-bt_k}] - a(1 - e^{-bt_k}) - \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!], \tag{3.2.13}$$

となる. したがって, 式 (3.2.13) の対数尤度関数から,

$$\frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial b} = 0, \tag{3.2.14}$$

を整理して

$$a = \frac{y_n}{(1 - e^{-bt_n})}, \tag{3.2.15}$$

$$\frac{y_n t_n e^{-bt_n}}{(1 - e^{-bt_n})} = \sum_{k=1}^{n} \frac{(y_n - y_{n-1})(t_n e^{-bt_n} - t_{n-1} e^{-bt_{n-1}})}{(e^{-bt_{n-1}} - e^{-bt_n})}, \tag{3.2.16}$$

となる. 式 (3.2.16) はパラメータ b のみに関係する方程式であるので, b について数値的に解いて b の推定値 $\hat{b}$ を求め, これを式 (3.2.15) に代入すれば a の推定値 $\hat{a}$ を得る. この最尤推定値 $\hat{a}$ および $\hat{b}$ を用いれば, 信頼性評価尺度の最尤推定値を得ることができる.

3.3 信頼性評価尺度

NHPP の平均値を用いて与えられるソフトウェアの信頼性評価尺度について述べておく.

3.3.1 ソフトウェア信頼度

テストが時刻 t まで進行しているときに, 時間区間 $(t, t+x]$ においてソフトウェア故障の発生しない確率は

$$R(x|t) = \exp[-\{H(t+x) - H(t)\}], \tag{3.3.17}$$

となる. 式 (3.3.17) は, ソフトウェア信頼度 (software reliability) と呼ばれる.

3.3.2 MTBF

時間区間 $(t, t+x]$ においてソフトウェア故障が発生する確率を求めると

$$F(x|t) = 1 - R(x|t) = 1 - \exp[-\{H(t+x) - H(t)\}], \tag{3.3.18}$$

により与えられるが, 平均値関数が $H(\infty) < \infty$ である場合は, $F(x|t)$ は通常の分布関数の条件 $F(\infty|t) = 1$ を満足していない. したがって, ソフトウェア故障発生時間間隔の平均値は通常の MTBF の定義通りには求められない. そのため, 実務的には NHPP の強度関数の逆数

$$MTBF_I(t) = \frac{1}{h(t)}, \tag{3.3.19}$$

や, テスト時間を NHPP の平均値関数で割った

$$MTBF_C(t) = \frac{t}{H(t)}, \tag{3.3.20}$$

を定義し, それぞれを平均ソフトウェア故障時間間隔あるいは平均フォールト発見時間間隔いわゆる MTBF の代替的尺度として考えることができる. これらはテスト時刻 t における瞬間 MTBF(instantaneous MTBF) および累積 MTBF(cumulative MTBF) と呼ばれている [3]. 本研究では, 実務でもよく用いられる累積 MTBF を採用し, 実測データに基づくソフトウェアの信頼性評価 (予測) について考察する.

4　メトリクスデータ

通常のソフトウェア開発では, ソフトウェア製品や開発プロセスを特徴付けるメトリクスデータとしてテストケース数や開発規模などが観測され, これらはフォールト検出事象に大きく影響を与えることが知られている [5]. 本研究の目的の一つは, 第 1 フェーズから第 $i-1$ フェーズまでのメトリクスデータを用いて, 最終（出荷直前）のフェーズである第 I フェーズのプログラム内に含まれたソフトウェアフォールト数を精度よく見積りたいというものである, 以下に, 文献 [1] に基づいて, あるソフトウェア開発プロジェクトにおいて観測されたメトリクスデータおよび時系列データの意味付けおよび変数の選択方法を示す.

4.1　変量の意味づけ

本研究では，既存の論文 [1] で使われたソフトウェア開発プロジェクトのデータを用いて，構築されたモデルの予測精度を評価する. 以下に，各データの意味付けも合わせて列挙する.

4.1.1　メトリクスデータ

各フェーズ $i(=1,2,\cdots,I)$ に対して品質に関連したメトリクス QM, 開発規模に関連したメトリクス SM が観測されている. 表 1, および表 2 に本研究で分析する 2 つのプロジェクトについての観測データを示す.

表 1　ProjectA のメトリクスデータ [1].

INC	No. days	No. faults	MD	UT	RV	No. test cases	Size	Effort
i	$t_{i,j}$	$x_{i,j}$	$y_{i,1}$	$y_{i,2}$	$y_{i,3}$	$y_{i,4}$	$y_{i,5}$	$y_{i,6}$
1	5	6	3	28	18	1259	24.1	78
2	7	14	2	25	25	686	22.3	118
3	4	3	4	6	7	664	8.6	53
4	4	4	1	19	10	797	6.6	70
5	3	2	2	24	16	673	10.2	67
6	5	8	3	26	19	1422	12.8	86

表 2　ProjectB のメトリクスデータ [1].

INC	No. days	No. faults	MD	UT	RV	No. test cases	Size	Effort
i	$t_{i,j}$	$x_{i,j}$	$y_{i,1}$	$y_{i,2}$	$y_{i,3}$	$y_{i,4}$	$y_{i,5}$	$y_{i,6}$
1	4	2	2	2	2	251	5.8	60
2	7	16	2	39	15	878	5.1	62
3	9	10	2	25	11	807	4.5	60
4	6	19	1	4	3	178	1.1	33
5	3	4	1	18	4	178	1.6	29
6	7	16	1	9	9	142	5.2	60

ここで, 記号は以下の通りである.

i　　INC

ソフトウェア開発プロジェクトにおけるフェーズ数.

t_i No. of days
第 i フェーズにおけるテストにかかった日数.

x_i No. of faults
第 i フェーズに発見された累積ソフトウェアフォールト数.

$y_{i,1}$ MD: No. of modules in development (0〜100)
第 i フェーズにおいて, 開発されたモジュールの数.

$y_{i,2}$ UT: No. faults detected in unit test
第 i フェーズの結合テストにおいて検出されたソフトウェアフォールト数.

$y_{i,3}$ RV: No. of reviews (日/KLOC)
第 i フェーズの結合テストの前に行われるレビューの数.

$y_{i,4}$ No. of test cases
第 i フェーズのシステムテストで実施されたテストケース数.

$y_{i,5}$ Size
第 i フェーズの開発規模の大きさ. KLOC で表される (1 KLOC=1000 Lines of Code).

$y_{i,6}$ Effort
第 i フェーズの延べのテスターの数.

4.1.2 時系列データ

提供されているデータは, 上述のメトリクスデータだけではなく, 時系列データも与えられている. これについて述べる. 各フェーズ $i(=1,2,\cdots,I)$ に対してのテスト実施日および発見フォールト数からなり，その総和がフェーズ i における x_i に一致する．表 3 および表 4 に本研究で分析する 2 つのプロジェクトについての観測データを示す． 例えば, 表 3 の第 1 フェーズの時系列データを図に示すと, 図 4 となる．これに対して，

表 3 ProjectA の時系列データ [1].

	INC.1		INC.2		INC.3		INC.4		INC.5		INC.6	
ST 期間	10/27	2	11/24	4	12/22	1	1/28	1	2/26	0	3/22	1
	10/28	0	11/25	1	12/23	0	1/29	3	2/27	2	3/23	3
	10/29	1	11/26	1	12/24	0	1/30	0	2/28	0	3/24	1
	10/30	2	11/27	3	12/25	2	1/31	0			3/25	1
	11/1	1	11/28	2	12/26						3/26	2
			11/29	1							3/27	0
			11/30	2								

前述した NHPP に基づくソフトウェア信頼度成長モデルを当てはめることとなる．

表 4　ProjectB の時系列データ [1].

	INC.1		INC.2		INC.3		INC.4		INC.5		INC.6	
ST 期間	4/28	1	6/9	3	7/11	0	8/4	1	8/30	1	9/23	4
	4/29	0	6/10	5	7/12	6	8/5	3	8/31	1	9/24	3
	4/30	0	6/11	0	7/13	2	8/6	0	9/1	2	9/26	0
	5/1	1	6/12	0	7/14	0	8/7	0	9/2	0	9/27	3
			6/13	4	7/15	1	8/8				9/28	3
			6/14	4	7/16	1	8/9				9/29	2
			6/15	0	7/17	0	8/10				9/30	1
					7/18	0	8/11				10/1	0
					7/19	0	8/12					

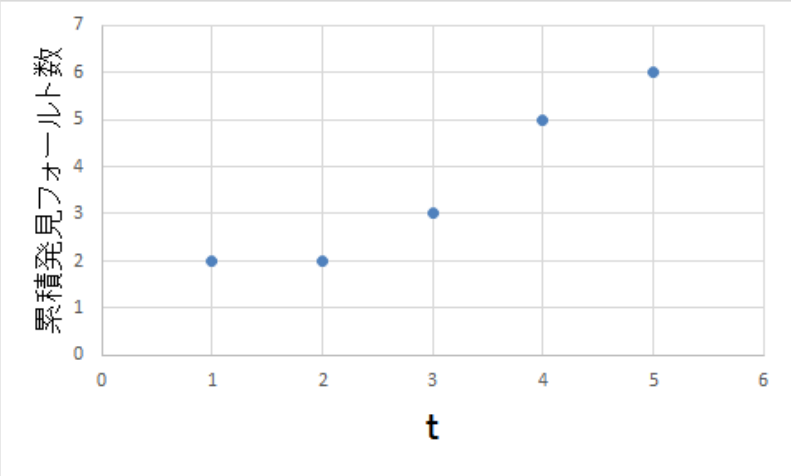

図 4　ProjectA の第 1 フェーズの時系列データ.

5　モデルの適用法

5.1　適用手順の概要

本研究では，以下の手順に従って，インクリメンタル開発の最終フェーズ（その番号を I で表す）における累積 MTBF を推定する方法を提案する．実際の詳細については次節に述べることとし，ここでは概略を説明しておく．

Step 1： フェーズ番号を $i = 1$ とする．

Step 2： 第 i フェーズでの時系列データに対して，NHPP モデルを適用する．平均値関数 $H(t)$ については，指数形あるいは遅延 S 字形を用いることとなるが，その選択については時系列データの時間変化の振舞いをグラフに描き，その大まかな形状から選択する．このような平均値関数の選択は，ウォーターフォール型開発での NHPP モデルの適用においても通常なされる方法である．その結果として，NHPP の適用により，そのフェーズの平均値関数に含まれるパラメータ，a_i および b_i が推定される．このときの推定法は，本研究での試みとして，最尤法と最小二乗法のそれぞれを試すこととする．

Step 3： $i = i + 1$ として，もし $i < I - 1$ ならば Step 2 に戻る．

以上により，信頼性予測の対象であるフェーズ I に対して，$I - 1$ までの時系列解析が終了している．続いて以下に従う．

Step 4： ここまでに得られた (a_i, b_i) $(i = 1, 2, \ldots I - 1)$ を用いて，以下の構造を仮定した上でメトリクス情報を利用して，(a_i, b_i) の値を修正した上で，更に第 I フェーズの予測に用いるためのパラメータ，(a_I, b_I) を推定する．

つまり，各フェーズで得られた (a_i, b_i) の推定値は，前者はそのフェーズでの潜在フォールトの数，後者はフォールトの発見率に対応する量である．それらはそのフェーズでの開発作業によって値の大小が影響されると考えるわけである．したがって，定数パラメータ α_0〜α_3，および β_0〜β_3 を導入して

$$\log a_i = \log \alpha_0 + \alpha_1 y_{i,k} + \alpha_2 y_{i,l} + \alpha_3 y_{i,m}, \tag{5.1.1}$$
$$\log b_i = \log \beta_0 + \beta_1 y_{i,k} + \beta_2 y_{i,l} + \beta_3 y_{i,m} \quad (i = 1, 2, \ldots, I - 1), \tag{5.1.2}$$

の関係があると仮定する．この構造は先行研究 [1] において採用されたものを修正して用いている．式 (5.1.1) および式 (5.1.2) は重回帰モデルであり，目的変数と説明変数の候補から，もっとも当てはまりの良い説明変数 y_* を 3 つ選択する．その後，最小二乗法によってこれら重回帰モデルの未知パラメータを推定する [6].

Step 5： 推定された未知パラメータ，$\hat{\alpha}_0$〜$\hat{\alpha}_3$，および $\hat{\beta}_0$〜$\hat{\beta}_3$ を用いて，フェーズ 1 からフェーズ $I-1$ の平均値関数のパラメータ a_i および b_i を修正することができる．さらに，当初の目的であるフェーズ I の NHPP の平均値関数に含まれるパラメータ a_I および b_I を，フェーズ I の時系列が得られるより前に推定することが可能となる．つまり，

$$\log a_i = \log \hat{\alpha}_0 + \hat{\alpha}_1 y_{i,k} + \hat{\alpha}_2 y_{i,l} + \hat{\alpha}_3 y_{i,m}, \tag{5.1.3}$$
$$\log b_i = \log \hat{\beta}_0 + \hat{\beta}_1 y_{i,k} + \hat{\beta}_2 y_{i,l} + \hat{\beta}_3 y_{i,m} \quad (i = 1, 2, \ldots, I), \tag{5.1.4}$$

により，左辺の a_i，b_i はそれぞれ更新かつ推定される．

これにより，最終フェーズでの平均値関数の振舞いを規定するパラメータが予測可能となることから，第 $I-1$ フェーズが終了し，第 I フェーズのコーディング・レビュー過程が終了した時点において，つまり最終テストをする前の段階で，最終テストでどのようなフォールトの発見過程となるかを予測することができる．これは従来の単純な時系列解析モデルをそのままインクリメンタル開発に適用しただけでは不可能な，新しい推定法と言える．次が最終ステップとなる．

> Step 6： a_I および b_I を用いて，フェーズ I での平均値関数 $H_I(t)$ が得られる．これを用いて，累積 MTBF を描画し，最終フェーズ I における信頼性評価・予測として用いる．

以上が本研究で開発した，インクリメンタル開発向けの信頼性予測手順である．

5.2 データ解析例

構築した信頼性評価モデルを実測値に当てはめモデルの評価を行う．第 1 フェーズから第 $I-1$ フェーズまでのメトリクスデータを使い最小二乗法, 最尤法により NHPP モデルの未知パラメータを推定する．その後, メトリクス情報を用いたモデルに適用し, 第 I フェーズのメトリクスデータを用いて推定を行う．さらに MTBF の推定値と MTBF の実測値との比較を行い評価を行う．
ここでは, 最小二乗法, 最尤法ともに ProjectA を例に解析を行う手順を示す.

5.2.1 最小二乗法

はじめに, 最小二乗法を用いた信頼性評価モデルの説明をする.
Step 1：

フェーズ番号を $i=1$ とする

Step 2： a_i, b_i の推定

各フェーズ i $(i=1,2,\cdots,I)$ のフォールト数の期待値を a_i, フォールト発見率を b_i, 時刻を t $(=0,1,\cdots,T)$, 時刻 t までに発見された累積ソフトウェアフォールト数を $x_{i,t}$ とする.
各フェーズごとに

$$H_i(t) = a_i[1-(1+b_it)e^{-b_it}], \tag{5.2.5}$$

$$sse_i = \sum_{t=0}^{T}(x_{i,t}-H_i(t))^2, \tag{5.2.6}$$

を設定し, 最小二乗法により第 $I-1$ フェーズまでのフォールト数の期待値 a_i , フォールト発見率 b_i, を推定する.
ここで, a_i, b_i ともに対数をとる．a_i, b_i は表 5, 表 6 のようになる.

Step 3：
$i=i+1$ として，もし $i<I-1$ ならば Step 2 に戻る

Step 4： 未知パラメータの推定
すべてのメトリクスデータを使うのではなく, 使用する変数の選択をする. 選択方法として $\log\hat{a}_i$,$\log\hat{b}_i$ を目的

表 5　最小二乗法により推定した $\hat{a}_i$, $\log \hat{a}_i$.

i	$\hat{a}_i$	$\log \hat{a}_i$
1	8.589	2.150
2	15.019	2.709
3	5.995	1.791
4	4.421	1.486
5	3.288	1.190

表 6　最小二乗法により推定した $\hat{b}_i$, $\log \hat{b}_i$.

i	$\hat{b}_i$	$\log \hat{b}_i$
1	0.474	-0.746
2	0.527	-0.640
3	0.351	-1.048
4	1.322	0.279
5	0.763	-0.270

変数とし, 説明変数 $y_{i,1}$〜$y_{i,6}$ から寄与率の高い変数の組み合わせを選択する本研究では寄与率の高い 4 組の組み合わせを使用する. 寄与率の高い組み合わせは表 7, 表 8 となる.

表 7　$\log \hat{a}_i$ の寄与率.

$y_{i,k}$, $y_{i,l}$, $y_{i,m}$	寄与率
$y_{i,1}$, $y_{i,2}$, $y_{i,5}$	0.992
$y_{i,2}$, $y_{i,5}$, $y_{i,6}$	0.978
$y_{i,2}$, $y_{i,4}$, $y_{i,6}$	0.961
$y_{i,1}$, $y_{i,3}$, $y_{i,6}$	0.967

表 8　$\log \hat{b}_i$ の寄与率.

$y_{i,k}$, $y_{i,l}$, $y_{i,m}$	寄与率
$y_{i,1}$, $y_{i,4}$, $y_{i,5}$	0.996
$y_{i,1}$, $y_{i,3}$, $y_{i,4}$	0.958
$y_{i,1}$, $y_{i,2}$, $y_{i,5}$	0.955
$y_{i,1}$, $y_{i,4}$, $y_{i,6}$	0.953

$\log \hat{b}_i$,$\log \hat{b}_i$ ともに寄与率の高い 4 組を
最小二乗法

$$sse_a = \sum_{i=1}^{I-1} (\log a_i - (\log \alpha_0 + \alpha_1 y_{i,k} + \alpha_2 y_{i,l} + \alpha_3 y_{i,m}))^2, \tag{5.2.7}$$

$$sse_b = \sum_{i=1}^{I-1} (\log b_i - (\log \beta_0 + \beta_1 y_{i,k} + \beta_2 y_{i,l} + \beta_3 y_{i,m}))^2, \tag{5.2.8}$$

により, パラメータ (α_0, α_1, α_2, α_3, β_0, β_1, β_2, β_3) を推定する.

Step 5： 第 I フェーズの $\hat{a}_i$, $\hat{b}_i$ の推定
推定した定数パラメータを

$$\log a_i = \log \alpha_0 + \alpha_1 y_{i,k} + \alpha_2 y_{i,l} + \alpha_3 y_{i,m}, \tag{5.2.9}$$

$$\log b_i = \log \beta_0 + \beta_1 y_{i,k} + \beta_2 y_{i,l} + \beta_3 y_{i,m}, \tag{5.2.10}$$

にあてはめ第 I フェーズのフォールト総数の期待値 $\hat{a}_I$, フォールト発見率 $\hat{b}_I$ を求める. 推定した定数パラメータ, 第 I フェーズのフォールト総数の期待値 $\hat{a}_I$, フォールト発見率 $\hat{b}_I$ は表 9, 表 10 になる.
Step 6： MTBF 実測値との比較
第 I フェーズの累積 MTBF の推定を行う. これに対応する実測値は $\frac{t}{x}$ で与えられる. 第 I フェーズのフォールト総数の期待値 $\hat{a}_I$, フォールト発見率 $\hat{b}_I$ を

表 9　第 I フェーズの $\hat{a}_I$.

α_0	0.373
α_1	0.308
α_2	-0.048
α_3	0.037
$\hat{a}_I$	8.962

表 10　第 I フェーズの $\hat{b}_I$.

β_0	1.485
β_1	-0.315
β_2	0.016
β_3	-0.032
$\hat{b}_I$	0.587

指数形 MTBF

$$\begin{aligned} MTBF &= \frac{t}{H(t)} \\ &= \frac{t}{a_I(1-e^{-b_I t})}, \end{aligned} \tag{5.2.11}$$

または
遅延 S 字形 MTBF

$$\begin{aligned} MTBF &= \frac{t}{H(t)} \\ &= \frac{t}{a_I[1-(1+b_I t)e^{-b_I t}]}, \end{aligned} \tag{5.2.12}$$

に代入し推定し, 下側 95% 信頼区間を求める. この信頼区間はポアソン分布に基づいて導出することができる. 例えば, 遅延 S 字形で第 I フェーズの実測値と推定値を比較すると図 5 のようになる.

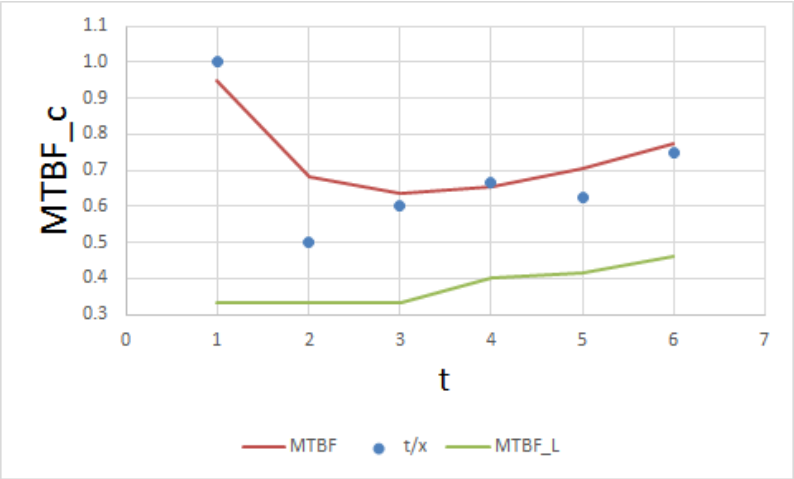

図 5　ProjectA の最小二乗法による遅延 S 字形信頼度成長モデルによる予測結果.

図より，第 I フェーズの時系列データが得られるより前に，MTBF の推定が概ね行えていることが見て取れる.

5.2.2 最尤法

続いて, 最尤法を用いた信頼性評価モデルの推定手順を説明する.

Step 1：

フェーズ番号を $i = 1$ とする

Step 2： a_i, b_i の推定

各フェーズ $i(= 1, 2, \cdots, I)$ のフォールト数の期待値を a_i, フォールト発見率を b_i とする. フェーズ i においては時系列データが (t_s, y_s) として与えられているとする. テスト時刻 t までに発見されたフォールト数が y_s である $(s = 0, 1, \cdots, n)$. また $t_0 = 0$, $y_0 = 0$ とする. このとき各フェーズごとに最尤法により時系列データの解析を行う. 平均値関数ごとに対数尤度方程式を以下に示す. まず指数形信頼度成長モデルの場合は下記の連立非線形方程式となる.

$$a_i = \frac{y_n}{(1 - e^{-b_i t_n})}, \tag{5.2.13}$$

$$\frac{y_n t_n e^{-b_i t_n}}{(1 - e^{-b_i t_n})} = \sum_{s=1}^{n} \frac{(y_s - y_{s-1})(t_s e^{-b_i t_s} - t_{s-1} e^{-b_i t_{s-1}})}{(e^{-b_i t_{s-1}} - e^{-b_i t_s})}, \tag{5.2.14}$$

$$H_i(t) = a_i(1 - e^{-b_i t}), \tag{5.2.15}$$

一方, 遅延 S 字形信頼度成長モデルは以下の通りである.

$$a_i = \frac{y_n}{[1 - (1 + b_i t_n)e^{-b_i t_n}]}, \tag{5.2.16}$$

$$\frac{y_n t_n^2 e^{-b t_n}}{[1 - (1 + b t_n)e^{-b t_n}]} = \sum_{s=1}^{n} \frac{(y_k - y_{s-1})(t_s^2 e^{-b t_s} - t_{s-1}^2 e^{-b t_{s-1}})}{\{(1 + b t_s)e^{-b t_{s-1}} - (1 + b t_s)e^{-b t_s}\}}, \tag{5.2.17}$$

$$H_i(t) = [1 - (1 + b_i t)e^{-b_i t}], \tag{5.2.18}$$

これらのいずれかにより, 第 $I - 1$ フェーズまでの, フォールト数の期待値 a_i, フォールト発見率 b_i, を求める. 平均値関数の選択方法については解析者の経験などで決定することとなる.

a_i,b_i ともに対数をとると, a_i, b_i は表 11, 表 12 のようになる.

Step 3：

表 11 最尤法により推定した $\hat{a}_i$ と $\log \hat{a}_i$.

i	$\hat{a}_i$	$\log \hat{a}_i$
1	6.853	1.925
2	17.050	2.836
3	4.674	1.542
4	4.289	1.456
5	2.488	0.911

表 12 最尤法により推定した $\hat{b}_i$ と $\log \hat{b}_i$.

i	$\hat{b}_i$	$\log \hat{b}_i$
1	0.723	-0.325
2	0.522	-0.649
3	0.685	-0.379
4	0.674	-0.395
5	1.007	0.007

$i = i + 1$ として，もし $i < I - 1$ ならば Step 2 に戻る

Step 4： 未知パラメータの推定

最小二乗法による推定方法と同じく, すべてのメトリクスデータを使うのではなく, 使用する変数の選択をする. 選択方法として $\log \hat{a}_i$, $\log \hat{b}_i$ を目的変数とし, 説明変数 $y_{i,1}$〜$y_{i,6}$ から寄与率の高い変数の組み合わせを選択する. こちらも, 寄与率の高い 4 組の組み合わせを使用する. 寄与率の高い組み合わせは表 13, 表 14 となる.

表 13　$\log \hat{a}_i$ の寄与率.

$y_{i,k}$, $y_{i,l}$, $y_{i,m}$	寄与率
$y_{i,2}$, $y_{i,4}$, $y_{i,6}$	0.998
$y_{i,1}$, $y_{i,2}$, $y_{i,5}$	0.988
$y_{i,3}$, $y_{i,5}$, $y_{i,6}$	0.968
$y_{i,1}$, $y_{i,3}$, $y_{i,6}$	0.964

表 14　$\log \hat{b}_i$ の寄与率.

$y_{i,k}$, $y_{i,l}$, $y_{i,m}$	寄与率
$y_{i,1}$, $y_{i,2}$, $y_{i,5}$	0.995
$y_{i,3}$, $y_{i,5}$, $y_{i,6}$	0.982
$y_{i,1}$, $y_{i,3}$, $y_{i,6}$	0.960
$y_{i,2}$, $y_{i,4}$, $y_{i,6}$	0.936

$\log \hat{b}_i$, $\log \hat{b}_i$ ともに寄与率の高い 4 組を最小二乗法 (5.2.7), 式 (5.2.8) により, 定数パラメータ (α_0, α_1, α_2, α_3, β_0, β_1, β_2, β_3) を推定する.

Step 5： 第 I フェーズの $\hat{a}_i$, $\hat{b}_i$ の推定

推定した定数パラメータを式 (5.2.9), 式 (5.2.10) にあてはめ第 I フェーズのフォールト総数の期待値 $\hat{a}_I$, フォールト発見率 $\hat{b}_I$ を求める. 推定したパラメータ, 第 I フェーズのフォールト総数の期待値 $\hat{a}_I$, フォールト発見率 $\hat{b}_I$ を表 15, 表 16 にまとめる.

表 15　第 I フェーズの $\hat{a}_I$.

α_0	0.198
α_1	0.279
α_2	-0.078
α_3	0.050
$\hat{a}_i$	7.738

表 16　第 I フェーズの $\hat{b}_I$.

β_0	1.524
β_1	0.032
β_2	-0.001
β_3	-0.013
$\hat{b}_i$	0.550

Step 6： MTBF 実測値との比較

累積 MTBF の推定を行う. 対応する実測値は $\frac{t}{x}$ で与えられる. 第 I フェーズのフォールト総数の期待値 $\hat{a}_I$, フォールト発見率 $\hat{b}_I$ を指数形 MTBF 式 (5.2.11) または遅延 S 字形 MTBF 式 (5.2.12) に代入し推定し, 下側 95% 信頼区間を求める. 信頼区間の導出はポアソン分布に基づいた. 結果として, 例えば, 遅延 S 字形で第 I フェーズの実測値と推定値を比較すると図 6 のようになる.

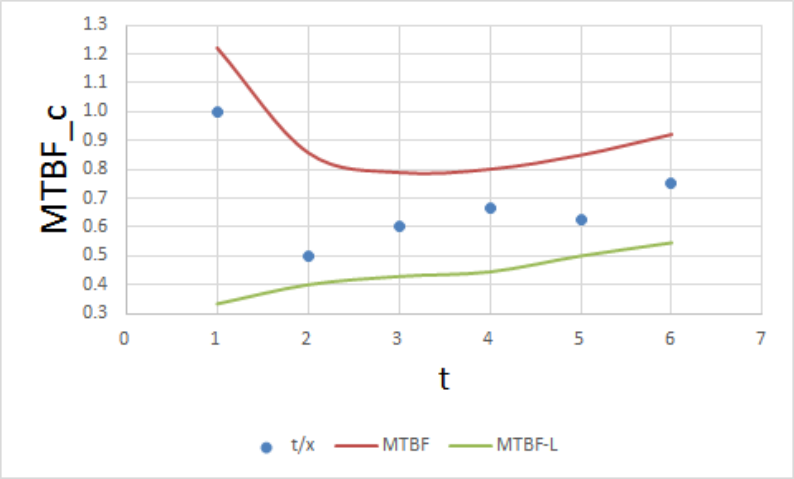

図 6　ProjectA の最尤法による遅延 S 字形信頼度成長モデル予測結果.

図より，こちらの場合も概ね MTBF の推定が行えていることがわかる．以上，NHPP モデルの当てはめについて，最小二乗法および最尤法の両方を試したが，それらのどちらを用いるべきかについては次章に述べる．

6　結果と考察

ProjectA, ProjectB ともに最小二乗法, 最尤法により解析し, 下側 95% 信頼区間の導出および第 I フェーズの実測値と推定値を比較した.

6.1　最小二乗法

はじめに, 最小二乗法による推定の結果は以下のようになった.

6.1.1　ProjectA

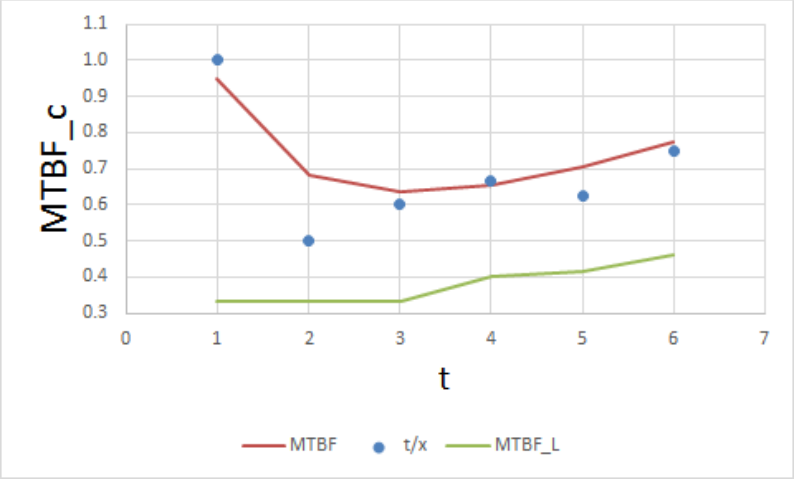

図 7　ProjectA の最小二乗法による遅延 S 字形信頼度成長モデル予測結果.

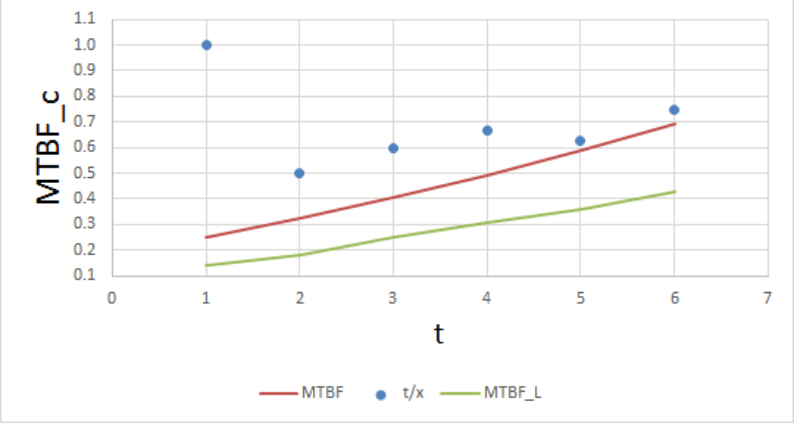

図 8　ProjectA の最小二乗法による指数形信頼度成長モデル予測結果.

最小二乗法を用いる場合，遅延 S 字形モデルを採用した方が, 実測値の振る舞いをよく表していることが分かる.

6.1.2 ProjectB

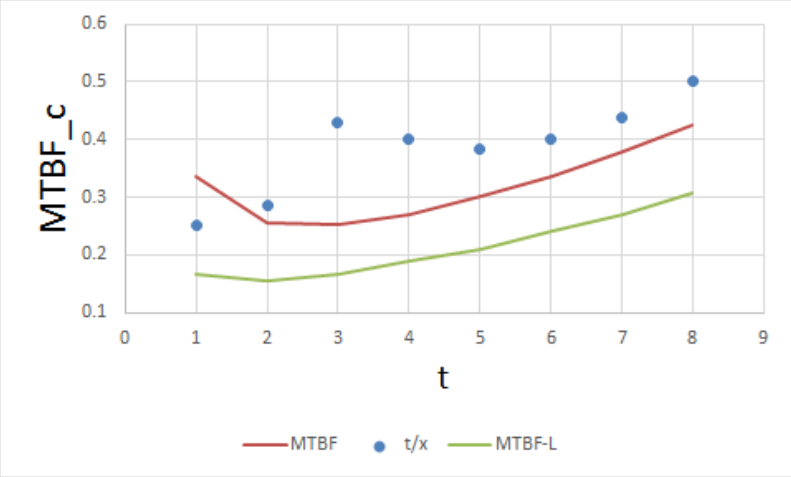

図 9　ProjectB の最小二乗法による遅延 S 字形信頼度成長モデル予測結果.

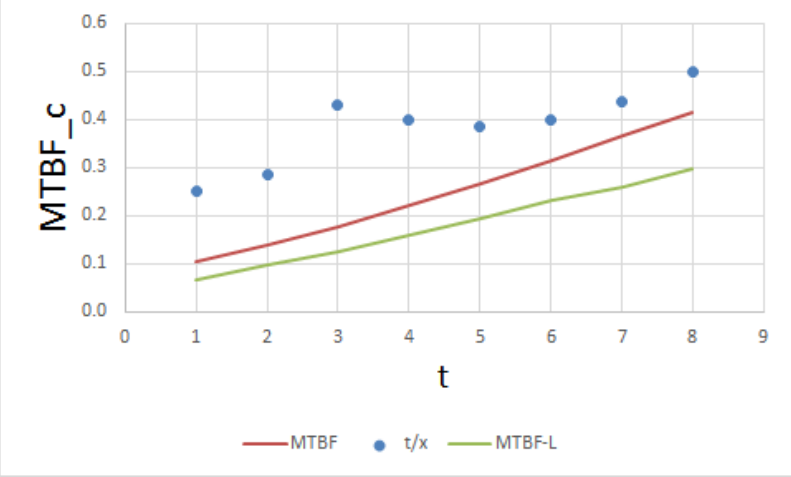

図 10　ProjectB の最小二乗法による指数形信頼度成長モデル予測結果.

Project B についても同様であり, 遅延 S 字形モデルを用いる方が概ね良い予測となっている.

6.2 最尤法

続いて 最尤法による推定の結果は以下のようになった.

6.2.1 ProjectA

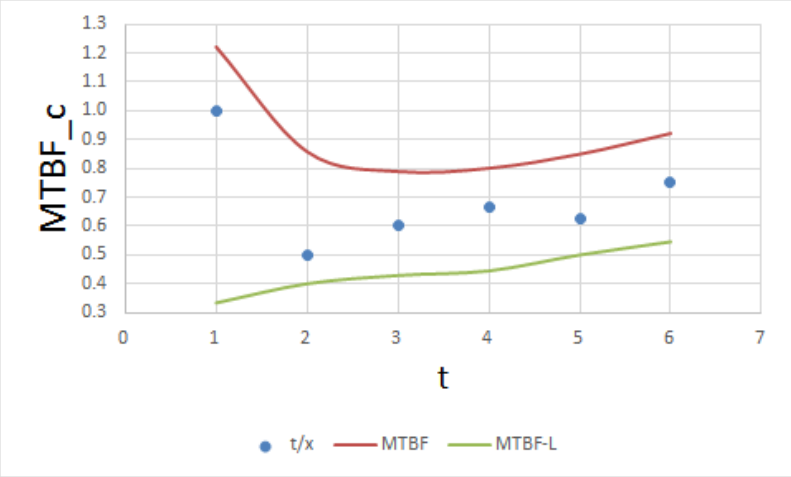

図 11　ProjectA の最尤法による遅延 S 字形信頼度成長モデル予測結果.

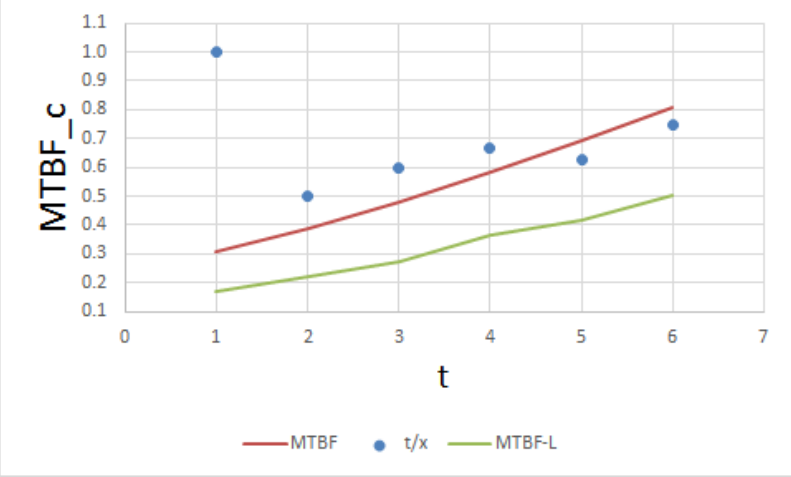

図 12　ProjectA の最尤法による指数形信頼度成長モデル予測結果.

指数形モデルを採用すると, 累積 MTBF の初期の振る舞いを表せていないことが見て取れる.

6.2.2 ProjectB

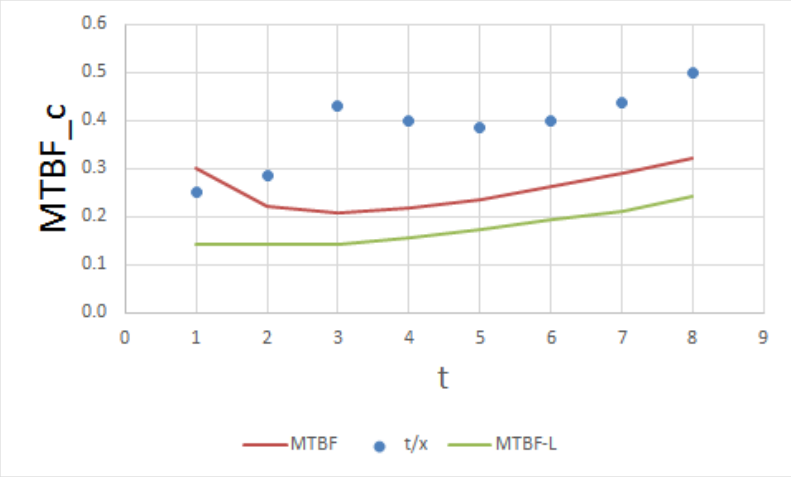

図 13　ProjectB の最尤法による遅延 S 字形信頼度成長モデル予測結果.

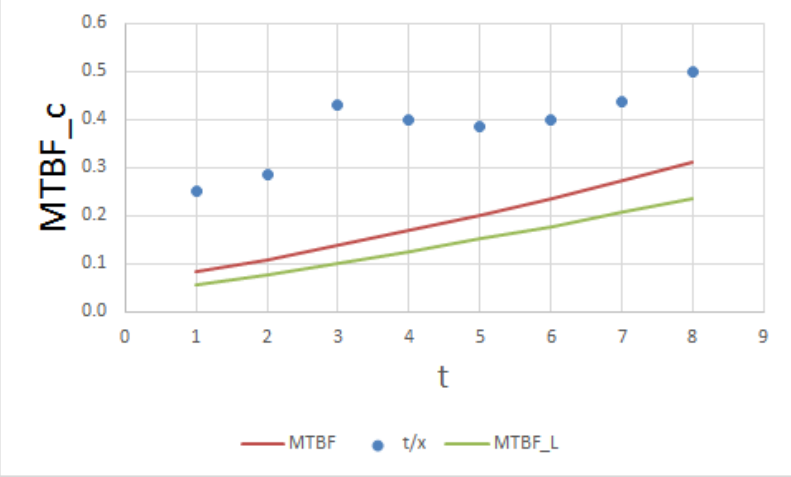

図 14　ProjectB の最尤法による指数形信頼度成長モデル予測結果.

Project B についても, A の場合の振る舞いと同様な結果となった.

また, ProjectA, ProjectB の各推定方法による残差平方和は表 17, 表 18 のようになった.

表より，Project A および B の両者において，累積 MTBF の予測推定精度の意味からは, これまで一般に用

表 17　ProjectA の各推定方法による残差平方和.

推定方法	残差平方和
最小二乗法遅延 S 字形	0.044
最小二乗法指数形	0.665
最尤法遅延 S 字形	0.311
最尤法指数形	0.524

表 18　ProjectB の各推定方法による残差平方和.

推定方法	残差平方和
最小二乗法遅延 S 字形	0.068
最小二乗法指数形	0.160
最尤法遅延 S 字形	0.184
最尤法指数形	0.322

いられてきた NHPP の未知パラメータを最尤法で推定するよりも, 最小二乗法でそれを行った方がよい結果となった. 遅延 S 字形モデルと指数形モデルの第 I フェーズでの選択については, 各プロジェクト特性などから, 評価者の経験などで決定せざるを得ないが, これは本研究において残された研究課題の一つである.

6.3 構造を考慮した信頼性評価法

本節では, ソフトウェアのインクリメンタル開発の構造を考慮し, インクリメントされた部分の潜在ソフトウェアフォールト数の推定やインクリメント部分のバグ除去率について考える. この信頼性評価法は, NHPP モデルの推定として, 最小二乗法, あるいは最尤法のいずれかの結果を用いて, 以下の手順に従う..

ここでは, ProjectA を最小二乗法で推定したものを例として説明する. 第 i フェーズに発見されたソフトウェアフォールト数 x_i, 第 i フェーズの潜在ソフトウェアフォールト数の期待値を a_i とすると表 19 となる.

表 19 ProjectA の x_i, a_i.

i	x_i	a_i
1	6	7.004
2	4	16.053
3	3	6.413
4	4	4.125
5	2	3.780
6	8	8.962

6.3.1 インクリメントされた部分の潜在ソフトウェアフォールト数の推定値

第 i フェーズにインクリメントされた部分の潜在ソフトウェアフォールト数の推定値を n_i とすると

$$n_1 = a_1, \tag{6.3.1}$$
$$n_2 = a_2 - (a_1 - x_1), \tag{6.3.2}$$
$$n_3 = a_3 - (a_2 - x_1), \tag{6.3.3}$$
$$n_4 = a_4 - (a_3 - x_3), \tag{6.3.4}$$
$$n_5 = a_5 - (a_4 - x_4), \tag{6.3.5}$$
$$n_6 = a_6 - (a_5 - x_5), \tag{6.3.6}$$

となる. Project A のデータを用いて第 i フェーズのインクリメント部分の潜在ソフトウェアフォールト数 n_i を推定すると表 20 となる.

表 20　潜在ソフトウェアフォールト数の推定値 n_i.

i	n_i
1	7.004
2	15.049
3	4.360
4	0.712
5	3.655
6	7.182

6.3.2　インクリメンタル開発でのバグ除去率

続いて, 各フェーズでのバグ除去率を考える. 例えば, 第 i フェーズのテスト行っているとする. このとき, 第 $i-1$ フェーズでのインクリメント部分に対するテストの比重は小さくなると考える. この比重を r とし r は $0 < r < 1$ とする. 第 i フェーズでのバグ除去率を p_i とすると各フェーズのインクリメント部分のバグ除去率, バグ数との関係を

$$n_1 p_1 = x_1, \tag{6.3.7}$$

$$n_1(1-p_1)rp_2 + n_2 p_2 = x_2, \tag{6.3.8}$$

$$\begin{aligned} &n_1(1-p_1)(1-rp_2)r^2p_3 \\ &+n_2(1-p_2)rp_3 \\ &+n_3p_3 = x_3, \end{aligned} \tag{6.3.9}$$

$$\begin{aligned} &n_1(1-p_1)(1-rp_2)(1-r^2p_3)r^3p_4 \\ &+n_2(1-p_2)(1-rp_3)r^2p_4 \\ &+n_3(1-p_3)rp_4 \\ &+n_4p_4 = x_4, \end{aligned} \tag{6.3.10}$$

$$\begin{aligned} &n_1(1-p_1)(1-rp_2)(1-r^2p_3)(1-r^3p_4)r^4p_5 \\ &+n_2(1-p_2)(1-rp_3)(1-r^2p_4)r^3p_5 \\ &+n_3(1-p_3)(1-rp_4)r^2p_5 \\ &+n_4(1-p_4)rp_5 \\ &+n_5p_5 = x_5, \end{aligned} \tag{6.3.11}$$

$$\begin{aligned} &n_1(1-p_1)(1-rp_2)(1-r^2p_3)(1-r^3p_4)(1-r^4p_5)r^5p_6 \\ &+n_2(1-p_2)(1-rp_3)(1-r^2p_4)(1-r^3p_5)r^4p_6 \\ &+n_3(1-p_3)(1-rp_4)(1-r^2p_5)r^3p_6 \\ &+n_4(1-p_4)(1-rp_5)r^2p_6 \\ &+n_5(1-p_5)rp_6 \\ &+n_6p_6 = x_6, \end{aligned} \tag{6.3.12}$$

として仮定することができる.

これを用いて，例えば，$r = 0.99$ で各フェーズを推定すると表 21 となる.

表 21　バグ除去率 p_i.

i	p_i
1	0.857
2	0.873
3	0.469
4	0.981
5	0.529
6	0.895

表より，フェーズ 3 および 5 のテスト作業に残存バグ数が多い可能性を示唆する結果となった．ただし，r の値が変われば評価結果は変わるが，本研究の結果として，何らかの情報から r を推定することはできなかった．これも今後の課題となる．

6.3.3　インクリメント部分の残存バグ率

また, インクリメント部分は通過したフェーズの数だけデバッグされる機会があることを考慮し, その残存バグ率を dp_1 から dp_6 とすると各フェーズのインクリメント部分の残存バグ率は

$$dp_1 = (1-p_1)(1-rp_2)(1-r^2p_3)(1-r^3p_4)(1-r^4p_5)(1-r^5p_6), \tag{6.3.13}$$
$$dp_2 = (1-p_2)(1-rp_3)(1-r^2p_4)(1-r^3p_5)(1-r^4p_6), \tag{6.3.14}$$
$$dp_3 = (1-p_3)(1-rp_4)(1-r^2p_5)(1-r^3p_6), \tag{6.3.15}$$
$$dp_4 = (1-p_4)(1-rp_5)(1-r^2p_6), \tag{6.3.16}$$
$$dp_5 = (1-p_5)(1-rp_2), \tag{6.3.17}$$
$$dp_6 = (1-p_6), \tag{6.3.18}$$

となる．$r = 0.99$ で各フェーズを推定すると表 22 となる．

表 22　残存バグ率 dp_i.

i	p_i
1	0.00002
2	0.0001
3	0.0008
4	0.002
5	0.05
6	0.1

以上の様に，ProjectA, ProjectB の最小二乗法, 最尤法による推定結果の潜在ソフトウェアフォールト数の推定値 n_i, バグ除去率 p_i, 残存バグ率 dp_i は表 23, 表 24, 表 25, 表 26 のようになった．また, すべて $r = 0.99$ とした．

累積 MTBF の推定において精度の高かったモデルと推定方法は, いずれのプロジェクトにおいても, 遅延 S 字形モデルを最小二乗法により推定したものであった．これに基づけば, 表 23 から, Project A については，

表 23　ProjectA の最小二乗法による n_i, p_i, dp_i.

i	n_i	p_i	dp_i
1	7.004	0.857	0.00002
2	15.049	0.873	0.0001
3	4.360	0.469	0.0008
4	0.712	0.981	0.002
5	3.655	0.529	0.05
6	7.182	0.895	0.1

表 24　ProjectA の最尤法による n_i, p_i, dp_i.

i	n_i	p_i	dp_i
1	5.603	1	0.00004
2	18.604	0.769	0.0002
3	0.789	0.605	0.001
4	2.010	1	0.001
5	2.848	0.701	0.05
6	6.884	1	0.1

表 25　ProjectB の最小二乗法による n_i, p_i, dp_i.

i	n_i	p_i	dp_i
1	1.158	1	0
2	15.229	1	0
3	15.183	0.736	0.0002
4	13.958	1	0
5	7.029	0.718	0.05
6	17.782	0.828	0.2

第 5，第 6 フェーズのインクリメント部分の残存バグ数が多いと推定され，また Project B でも同様となる．しかし後者のプロジェクトではインクリメント部分の残存バグ率が 0 と推定されているフェーズもあることが示された．

表 26　ProjectB の最尤法による n_i, p_i, dp_i.

i	n_i	p_i	dp_i
1	2.118	1	0
2	18.124	0.980	0.008
3	14.490	0.204	0.4
4	7.109	0.214	0.5
5	0	0.137	0.7
6	22.876	0.226	0.8

7　おわりに

本研究では, ソフトウェアのインクリメンタル開発に適したソフトウェア信頼性成長モデルを考案し, 既存の論文内で使われたソフトウェアメトリクスデータを用いて，最小二乗法, 最尤法よって構築されたモデルを複数提案した上で, 各モデルの有用性を評価した. 2 つのプロジェクトに対し, 有用性を評価したが, どちらのプロジェクトも, 最小二乗法で推定したモデルのほうが精度が高い結果を得ることができた. 今後の課題としては, フェーズ数が少ないプロジェクトに対しても精度を上げることが挙げられる. しかしながら，データが少ないため, 変数選択やパラメータ推定をうまく行うことができなかった他のプロジェクトには適用できなかった. 今後の課題としては, 他の変数選択法やパラメータ推定を行うなどが挙げられる. これらの課題に取り組んでいくことにより, インクリメンタル開発における信頼性評価技術の向上, ひいてはこれからの情報化社会の発展に貢献できると考えている.

参考文献

[1] T. Fujii, T. Dohi, and T. Fujiwara, “Towards quantitative software reliability assessment in incremental development processes.”, *Proceedings of the 33rd International Conference on Software Engineering*, pp. 41-50. ICSE ’11, (2011).

[2] Mint（経営情報研究会），図解でわかる　ソフトウェア開発のすべて，日本実業出版 (2000)

[3] 山田茂，ソフトウェア信頼性モデル，日科技連出版社 (1994).

[4] 木村光宏，藤原隆次, ソフトウェアの信頼性，日科技連出版社 (2011).

[5] 山田茂，福島利彦，品質志向ソフトウェアマネジメント，森北出版 (2007).

[6] 山田茂，木村光宏，高橋宗雄，TQM のための統計的品質管理，コロナ社 (1998)

謝辞

本研究，および本論文の執筆に関して，終始多大なご指導を頂きました木村光宏教授に心より感謝致します．また, いつも様々な助言や温かい言葉を頂いた木村研究室の先輩方，同輩ならびに後輩達に感謝の意を表します．とても恵まれた環境で研究をすることができました. 最後に, ここまで支えてくれた家族の方々にも感謝致します. 2014 年 2 月椎名紘伸