博　士　論　文

テスト空間依存型
ソフトウェア信頼度成長モデルと
その適合性評価に関する研究

2002年1月

藤　原　隆　次

あらまし

近年、様々な分野で重要な役割を担うソフトウェア製品は，ますます複雑化および大規模化の一途を辿っており，これらのソフトウェア製品をテストするために費やされる開発労力は，総開発労力の半分以上を占めていると言われる．このようなソフトウェア製品の高信頼化を図るための一手法として，ソフトウェア信頼度成長モデルによる様々な定量的信頼性評価技術が研究されている．また，実際のテスト工程におけるソフトウェアフォールト発見事象を，どのソフトウェア信頼度成長モデルを用いて記述し，ソフトウェア信頼性評価を行うかは，開発管理者にとって重要な問題である．ここで，これらのソフトウェア信頼度成長モデルにより，開発されたソフトウェア製品の期待残存フォールト数，ソフトウェア信頼度，および平均ソフトウェア故障時間間隔などの定量的な推定が可能になったとはいえ，いまだに開発者の能力に大きく依存しているのが現状である．なぜなら，テスト工程において，ソフトウェア信頼性の評価・管理を実施したソフトウェア製品で，ユーザ障害の頻発するケースが稀に観測されているからである．これは，今までに構築されたソフトウェア信頼度成長モデルには，フォールト発見事象に影響を及ぼすテスト要因が十分に考慮されていないためである．

また，ソフトウェア開発者は，テスト工程において，要求仕様書に基づいてインプリメントした機能を検証するために，多種・多様なテストケース（テスト用入力

データの集合体）を投入する．このとき，投入したテストケースにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体が形成される．この集合体をテスト空間と呼ぶ．ソフトウェアシステム内におけるテスト空間の占有率は，テストケースの投入量およびテスト要因と密接な関係をもっている．

そこで本研究では，フォールト発見事象を的確に表現するために，ソフトウェア開発のテスト工程において，投入されたテストケースにより形成されるテスト空間の時間的推移がフォールト発見事象に影響するものと仮定し，テストの人的要因としてテストケース設計者のテスト習熟性，およびフォールト修正時における新規フォールト混入可能性が，発見されるフォールト数に影響を及ぼすものと仮定したテスト空間依存型ソフトウェア信頼度成長モデルを提案する．これらのソフトウェア信頼度成長モデルは，非同次ポアソン過程により記述することができる．最後に，これらのソフトウェア信頼度成長モデルを実際に開発プロジェクトの現場で採取されたフォールトデータに適用した信頼性評価例を示し，幾つかの適合性評価基準を導入して，従来の非同次ポアソン過程に基づくソフトウェア信頼度成長モデルとの適合性比較を行う．

目　　次

図 一 覧

表　一　覧

第1章　まえがき

1.1　研究の背景と目的

今日の高度情報化社会において，様々な分野でコンピュータシステムが利用され，一般社会の中で重要な役割を担っている．このため，コンピュータシステムの高信頼化を図る問題において，主要な構成要素の一つであるハードウェアに関する分野は十分研究され高信頼性が実現されている現在，ソフトウェアシステムの信頼性の低さが問題視されるようになった．近年，ソフトウェアシステムは，大規模化・複雑化・多様化の一途を辿っており，その開発は多くの要員が参画して作成された知的生産物，つまりドキュメントおよびソースプログラムを成果物とする人間集約的作業である．そのため，ソフトウェアシステムの開発過程における欠陥や誤り，いわゆる**ソフトウェアフォールト** (**software fault**，以下フォールトと略す) の作り込みは回避不可能な問題である．さらに，フォールトにより引き起こされる**ソフトウェア故障** (**software failure**) が表面化することで，長時間に渡るシステムダウンが続くと，社会生活に及ぼす影響は計り知れないものとなる．ときには，人命にかかわる重大な事故を引き起こす場合も考えられる．このような現状の中で，より品質・信頼性の高いソフトウェア開発が重要な課題となってきている．

そこで，このような状況下で開発されるソフトウェアシステムの品質を把握するための特性要因が精力的に議論されている．Wulf [1] は，保全性，更新性，堅固性，

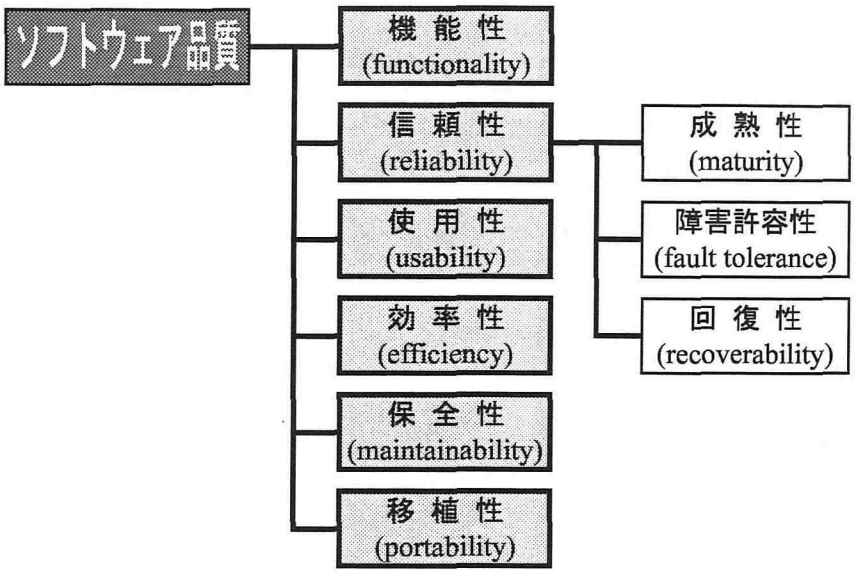

図 1.1: 標準的なソフトウェア品質特性 (ISO/IEC 9126).

明解さ，性能，移植性，操作性の7つの品質特性を取り上げた．その後，様々な考え方が提案され (例えば，Boehm [2] の品質モデル)，近年になってISO/IEC(国際標準合同委員会) で標準化されるようになった．この品質特性および副特性をまとめたのが図 1.1 である．この品質特性のうち，充足されていて当り前であり，不十分であるとユーザに不満足感を与えるという品質特性，すなわち「**当り前品質**」の特性に相当するのが**信頼性 (reliability)** であり，この確保が重要である．ここで，ソフトウェアの信頼性は，ソフトウェア内に欠陥や誤りがないこと，障害が発生しても外部には影響がなかったり，あるいは最小化できる特性と考えられるので，

「規定の環境下で，任意の期間中に，ソフトウェア故障を引き起こすことなく動作できる性質や度合」

と定義される動的品質を表している．この動的品質を表すソフトウェアの信頼性を

把握するために，ソフトウェア開発の最終工程であるテスト段階や実際の運用・保守段階において発見される累積フォールト数の時間的推移を，ソフトウェアの信頼度が成長していく過程として考え，確率・統計論に基づく**ソフトウェア信頼度成長モデル**(software reliability growth model，以降SRGMと略す) [3]–[7] による様々な定量的信頼性評価技術が積極的に研究されている．特に，日本では傾向曲線モデルや**非同次ポアソン過程**(nonhomogeneous Poisson process，以降NHPPと略す) モデルなどがよく知られている．代表的な傾向曲線モデルであるゴンペルツ曲線モデルは，従来より多数の日本企業のソフトウェア開発において，ソフトウェアの信頼性評価を行うため幅広く用いられてきた．一方NHPPモデルについては，様々なテスト環境や運用環境を反映したモデルが提案され，多くの実測フォールトデータを用いて，提案されたモデルの適合性および妥当性が検討されている．SRGMの代表的な研究として，Jelinski & Moranda [8]，Moranda [9]，Littlewood [10]，Musa [11]，Goel & Okumoto [12]，Yamada & Osaki [13]，Yamada et al. [14] などがある．

ソフトウェア信頼性評価を実施するために，実際のテスト工程におけるフォールト発見事象を記述するSRGMの選択が，開発管理者にとって重要な問題である．ここで，これらのSRGMにより，開発されたソフトウェアシステムの期待残存フォールト数，ソフトウェア信頼度および平均ソフトウェア故障時間間隔など，様々な定量的信頼性評価尺度を用いた推定が可能になったとはいえ，いまだに開発者の能力に大きく依存しているのが現状である．なぜなら，テスト工程において，ソフトウェア信頼性評価・管理を実施したソフトウェアシステムで，ユーザ障害の頻発するケースが稀に観測されているからである．そこで，ユーザ障害が頻発するソフトウェアシステムに対して既存のSRGMを適用して調査・分析してみると，何れも実測フォー

ルトデータに対する誤差が比較的大きいソフトウェアシステムが問題であることが判明した．これは，今までに構築されたSRGMには，フォールト発見事象に影響を及ぼすテスト要因が十分に考慮されていないことが原因と考えられる．したがって，フォールト発見事象を的確に表現するためのテスト要因として，

・テストケース設計者のテスト習熟性

・フォールト修正時における新規フォールト混入可能性

を取り上げる．

ソフトウェア開発者は，テスト工程において，要求仕様書に基づいてインプリメントした機能を検証するために，多種類および多数の**テストケース** (**test-case**，テスト用入力データの集合体) を投入する．このとき，投入したテストケースにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体が形成される．この集合体を**テスト空間** (**testing-domain**)[15],[16] と呼ぶ．ソフトウェアシステム内におけるテスト空間の占有率は，テストケースの投入量および上述のテスト要因と密接な関係をもつことは明白である．つまり，テスト習熟性の高低がテストケース品質に影響することで，ソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体の影響範囲が決定される．すなわち，フォールトの発見可能性に影響することから，テスト空間占有率はテストケースの投入量およびテストケース設計者のテスト習熟性に密接な関係をもつ．さらに，実際のデバッグ作業に注目すると，フォールト修正および除去作業がいつも確実に実施されるとは限らず，新規フォールトの作り込まれる可能性を考慮する方がより現実的である．

そこで本研究では，フォールト発見事象を的確に表現するために，ソフトウェア開発のテスト工程において，投入されたテストケースにより形成されるテスト空間の時間的推移がフォールト発見事象に影響するものと仮定し，上述のテスト要因がテストにおいて発見されるフォールト数に影響するものと仮定した幾つかのテスト空間依存型SRGMを提案する．

1.2 本論文の構成

本論文は，本章を含め全6章から構成されている．

第2章では，要求仕様書に基づいてインプリメントされた機能を検証するために投入されるテストケースにより影響を受けるモジュールおよび機能のテストパスに関する集合体，つまりテスト空間について議論する．さらに，ソフトウェアシステム内におけるテスト空間占有率の時間的推移に多大な影響を及ぼすテスト要因として，テスト対象ソフトウェアシステムに対するテストケース設計者の要求仕様書および内部構造の理解度を示す**テスト習熟性** (**testing-skill**)，およびフォールト修正時において新規フォールトが作り込まれる可能性を意味する**不完全デバッグ** (**imperfect debugging**) **環境**を考慮したテスト空間についても議論する．

第3章では，発見されたフォールト数を計測することにより記述できる計数過程に対して，簡潔でかつ実際のフォールト発見事象の意味付けが容易であるNHPPに基づくSRGMについて議論する．特に，ソフトウェアシステム内におけるテスト空間占有率の時間的推移が，フォールト発見事象に影響するものと仮定した4種類のテスト空間依存型SRGMを提案する．また，提案したSRGMに基づく信頼性評価尺度の導出，およびモデルパラメータの推定方法について記述する．さらに，適用

したフォールトデータに対する適合性を統計的に検定する方法についても記述する．

第4章では，第3章で議論したSRGMを実際に観測されたフォールトデータに適用し，その適用結果および定量的信頼性評価結果を示すと共に，幾つかの評価基準を用いて既存モデルとの適合性比較を行う．

第5章では，実際のソフトウェア開発現場における品質・信頼性管理の取り組みとして，一般的なテスト管理者および実務者が使用可能なCASEツールとして開発したソフトウェアテスト管理ツール，およびデータベースによるフォールト情報の一元管理，かつ全プロジェクトにおいて情報共有および認識が可能な新管理システムについて議論する．さらに，両者を連携させることにより得られた効果についても記述する．

第6章では，結論として本論文の全般的な総括を実施し，得られた結果を要約する．

第2章　テスト空間の記述

ソフトウェア開発者は，テスト工程において，要求仕様書に基づいてインプリメントした機能を検証するために，多種類および多数のテストケースを投入する．このとき，ソフトウェアシステム内には，投入されたテストケースにより影響を受けるモジュールおよび機能のテストパスに関する集合体，つまりテスト空間が形成される．テスト空間は，投入されるテストケースの増加と共に拡大し，これに伴って発見可能となるフォールト数も増加する．そして，最終的にはソフトウェアシステム全体にまで拡大するものと推測できる．

本章では，ソフトウェアシステム内におけるテスト空間の占有率が，

- テストケース設計者のテスト習熟性
- フォールト修正時における新規フォールト混入可能性

のようなテスト要因およびテストケース投入量と密接な関係をもつことに着目した４種類のテスト空間，つまり2.1節において基本形テスト空間，2.2節においてテストケース設計者のテスト習熟性を考慮したテスト空間，および2.3節においてフォールト修正時における不完全デバッグ環境を考慮したテスト空間について議論する．また，テスト習熟性を考慮したテスト空間において，フォールト発見可能領域の拡大率とテスト空間拡大率が近似的に同一の値をとるものと仮定した簡易形，および

両者の拡大する時期が異なることから同一の値にならないと仮定した一般形についても議論する．さらに，議論した4種類のテスト空間のソフトウェアシステム内における占有率と，テストにより発見可能となるフォールト数との関係を定式化する．最後に，導出された4種類のテスト空間関数に基づくそれぞれのテスト空間成長関数を導出する．

2.1　基本形テスト空間

本節では，テスト空間の基本的な概念について議論する．

ソフトウェア開発の最終工程であるテスト段階において，ソフトウェア開発者は，多種類かつ多数の**テストケース** (**test-case**，テスト用入力データの集合体) を用いて，要求仕様書に基づいてインプリメントされたソフトウェアシステム内に潜在するフォールトの発見および修正を行っている．このとき，投入されたテストケースから影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体が形成される．このテストパスに関する集合体を**テスト空間** (**testing-domain**)[15],[16] と呼ぶ (図2.1参照)．このテスト空間は，投入されるテストケースの増加と共に拡大し，これに伴って発見可能となるフォールト数も増加する．そして，最終的にソフトウェアシステム全体にまで拡大するものと推測できる．さらに，テスト空間内にフォールトが存在する場合，フォールトは投入されたテストケースにより何らかの影響を受け，その度合に従って出力結果にさらに影響を及ぼすことが考えられる．この出力結果を分析することにより，ソフトウェア故障か否かの判断を行う．

したがって，ソフトウェアシステム内におけるテスト空間占有率と，テストによ

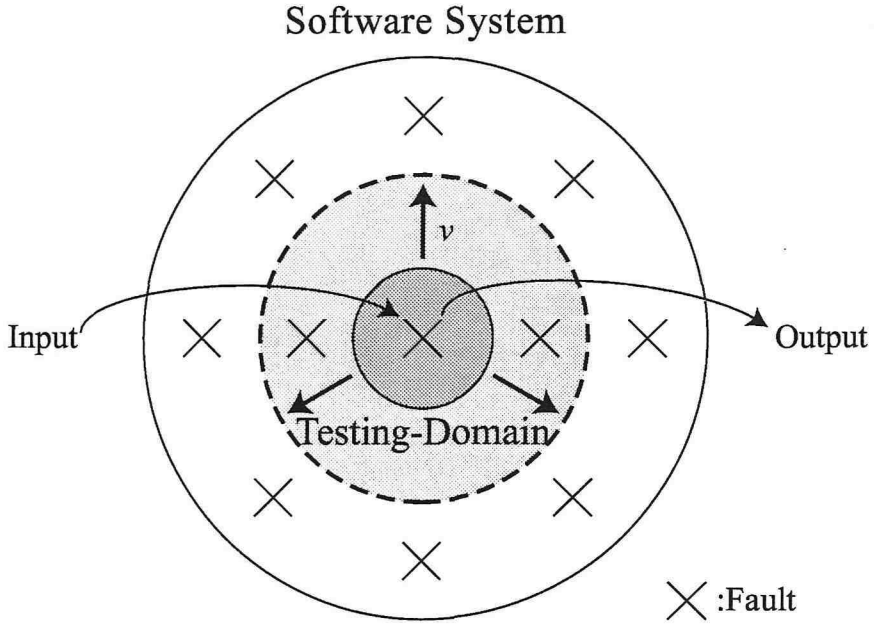

図 2.1: ソフトウェアシステム内の基本形テスト空間概念図.

り発見可能となるフォールト数との関係を定式化するために，次のような仮定を設定する [15],[16]：

1) フォールト修正により，新たなフォールトを作り込まない.

2) ソフトウェアシステム内に潜在するフォールトは，テスト空間内に一様に分布する.

3) 任意のテスト時刻 t において発見可能なフォールト数の増加率は，テスト空間外に残存する総フォールト数に比例する.

これらの仮定から，この事象を次の微分方程式で表すことができる.

$$\frac{du(t)}{dt} = v\left[a - u(t)\right] \ (a > 0,\ v > 0)\,. \tag{2.1}$$

式 (2.1) で使用しているパラメータおよび関数は

$a =$ テスト開始前にソフトウェアシステム内に潜在する総期待フォールト数,

$v =$ テスト空間拡大率,

$u(t) =$ 任意のテスト時刻 t においてテスト空間内に潜在する総フォールト数,

を表す．ゆえに，テスト開始時点において，テストケース設計者の潜在能力として発見可能なフォールト数は 0 個であり，テスト空間も存在しないことを意味する初期条件 $u(0) = 0$ の下で，微分方程式 (2.1) を $u(t)$ に関して解くことにより次式を得る．

$$u(t) \equiv u_a(t) = a\{1 - \exp[-vt]\}. \tag{2.2}$$

ここで，$\{1 - \exp[-vt]\}$ は，ソフトウェアシステム内におけるテスト空間占有率を

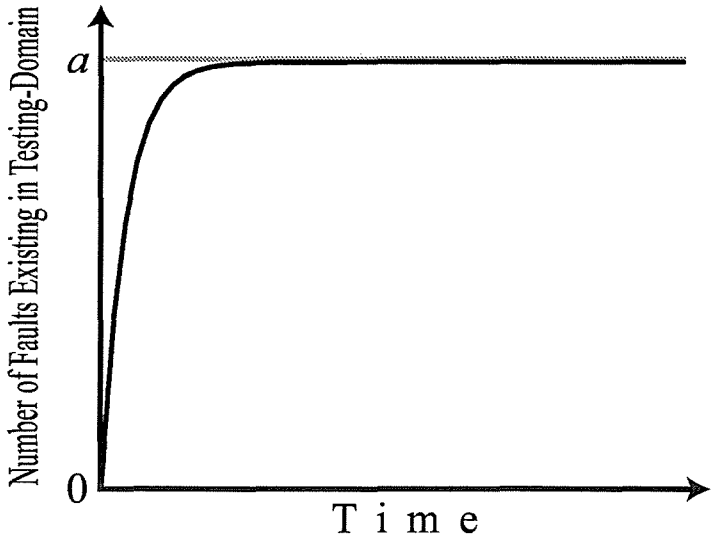

図 2.2: 基本形 T-D 関数 $u_a(t)$ の時間的推移.

表し，その占有率は指数形成長曲線に沿って拡大することを意味する (図2.2参照). また，$u(t)$ を**テスト空間関数 (testing-domain function)** と呼び，特に式(2.2) の $u_a(t)$ を**基本形テスト空間関数** (basic testing-domain function，以降**基本形 T-D 関数**と略す) と呼ぶ.

2.2 テスト習熟性を考慮したテスト空間

本節では，テストケースの品質に対して多大な影響を及ぼす一因と考えられるテストケース設計者のテスト習熟性について考える.

テストケース設計者の過去のテスト経験，あるいはテスト対象ソフトウェアシステムの要求仕様書および内部構造の理解度は様々であるため，テスト工程初期においてテストケース設計者のテスト習熟性が低い場合と高い場合が考えられる．そこで，テスト工程初期におけるテスト習熟性の高低の一例として，次のように考えることができる：

低い場合：テストケース設計者が新入社員など，テスト対象ソフトウェアシステムの要求仕様書および内部構造の理解度が低い人

高い場合：テストケース設計者が中堅社員など，テスト対象ソフトウェアシステムの要求仕様書および内部構造の理解度が高い人

ここで，テスト習熟性が向上する場合について考えてみよう．先ず，テストケースを設計する段階において，テスト対象ソフトウェアシステムの要求仕様書および内部構造を理解することによりテスト習熟性が向上する．さらに，テストのために設計したテストケースを投入することによりテスト習熟性が向上する．ゆえに，テスト習

熟性とはテストケース設計者の潜在能力，つまりテストケース設計者が発見可能となる総フォールト数を表す**フォールト発見可能領域 (fault-detection possibility)** により表現することができる (図 2.3 参照).

したがって，テスト習熟性が低い場合とは，テストケース設計者がテスト対象ソフトウェアシステムの要求仕様書および内部構造を十分理解できていない状態，つまりテストケース設計者がフォールトを発見できる可能性が低い状態であるため，設計されたテストケースがソフトウェアシステム内のモジュールおよび機能に影響を及ぼす範囲が狭く，必ずしもフォールトが発見できるとは限らない場合である．これより，テストケースの投入量の増加と共にテスト空間が必ずしも拡大しないことが考えられる．一方，テスト習熟性の高い場合とは，テストケース設計者がテスト対象ソフトウェアシステムの要求仕様書および内部構造を十分理解している状態，つま

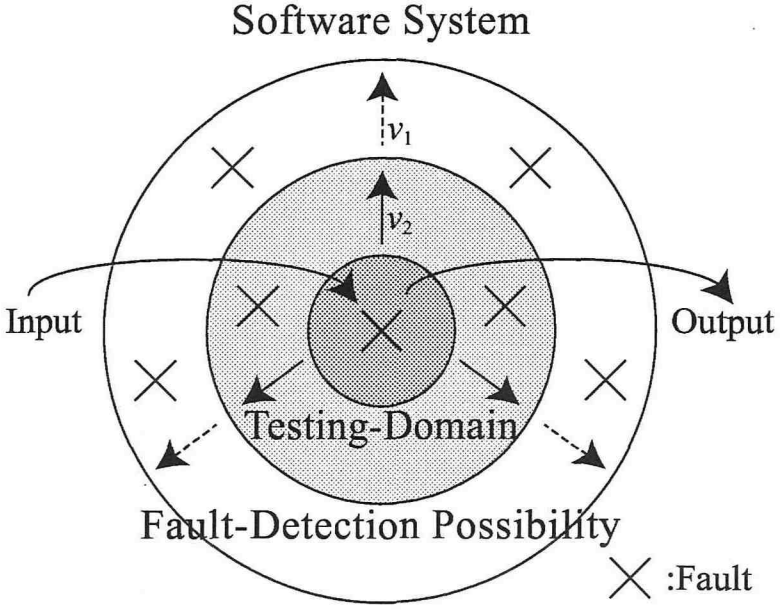

図 2.3: テスト習熟性を考慮したテスト空間概念図.

りテストケース設計者がフォールトを発見できる可能性が高い状態であることを表す．例えば，過去に類似ソフトウェアシステムのテスト経験を有するテストケース設計者であれば，要求仕様書および内部構造の理解度は高く，高品質なテストケース，つまりソフトウェアシステム内の広範囲のモジュールおよび機能に影響を及ぼすテストケースを設計することが可能である．さらに，このようなテストケースを投入することにより，効率的なテストが可能になり，少数のテストケースでテスト空間がソフトウェア全体にまで急速に拡大することが考えられる．

2.2.1 テスト習熟性を考慮したテスト空間の定式化

次に，ソフトウェアシステム内におけるテスト空間占有率と，テストによりテストケース設計者の発見可能となるフォールト数との関係を定式化する．定式化にあたり，前述の議論より

1) フォールト修正により，新たなフォールトを作り込まない．

2) 任意のテスト時刻 t において発見可能となるフォールト数の増加率は，フォールト発見可能領域外に残存するフォールト数に比例する．

3) 任意のテスト時刻 t においてテスト空間内に潜在するフォールト数の増加率は，テスト空間外に残存する発見可能な総フォールト数に比例する．

と仮定すれば，これらの事象を次の２つの微分方程式で表すことができる．

$$\frac{dw(t)}{dt} = v_1 [a - w(t)] \qquad (a > 0,\ v_1 > 0), \tag{2.3}$$

$$\frac{du(t)}{dt} = v_2 [w(t) - u(t)] \ (v_2 > 0). \tag{2.4}$$

ここで，各パラメータおよび関数は以下のことを表す．

$v_1 =$ フォールト発見可能領域の拡大率，

$v_2 =$ テスト空間拡大率，

$w(t) =$ 任意のテスト時刻tにおいて，テストケース設計者のテスト習熟性が向上することにより発見可能となる総フォールト数，つまり任意のテスト時刻tにおけるソフトウェアシステム内のフォールト発見可能領域．

2.2.2　簡易習熟性テスト空間関数

次に，微分方程式(2.3)および(2.4)を解くための初期条件について考える．さらにここでは，テストケース設計者のテスト習熟性が向上することにより発見可能となるフォールトの増加数，つまり単位テスト時間当りのフォールト発見可能領域の拡大率v_1と，テストケースを投入することにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体に潜在するフォールトの増加数，つまり単位テスト時間当りのテスト空間拡大率v_2が，近似的に$v = v_1 = v_2$としてテスト習熟性を考慮した簡易形のテスト空間関数を導出する．

先ず，テストケース設計者のテスト習熟性が低い場合について考えてみよう．この場合，前述したように新入社員などによりテストケースが設計されることから，テスト工程の初期段階におけるテストケース設計者のフォールト発見可能性が低いため，設計されたテストケースを投入してもソフトウェアシステム内におけるテスト空間の拡大率は小さい．つまり，投入したテストケースにより必ずしもフォールトが発見されるとは限らない．これは，テストケースの投入量の増加と共にテスト空

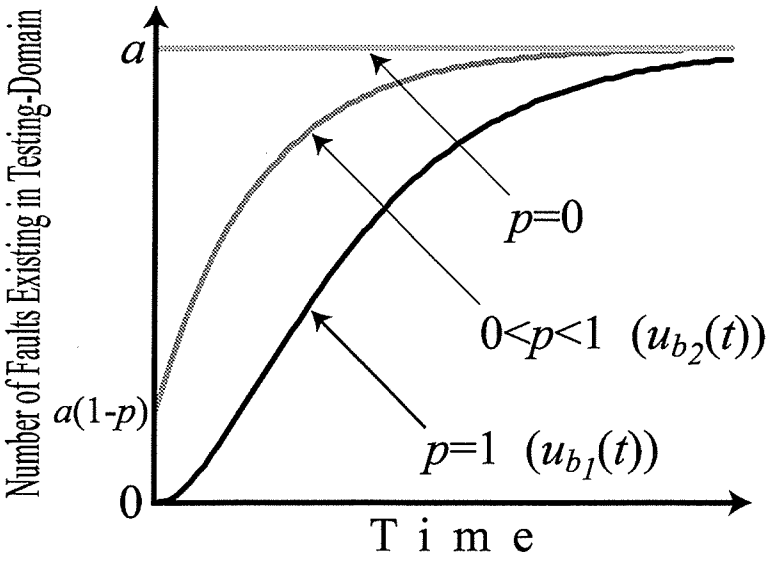

図 2.4: 習熟性 T-D 関数 $u_b(t)$ の時間的推移.

間が必ずしも拡大しないことを意味する. 以上のことを考慮して, 微分方程式 (2.3) および (2.4) を, それぞれ初期条件 $w(0) = 0$ および $u(0) = 0$ の下で $u(t)$ に関して解くことにより次式を得る.

$$u(t) \equiv u_{b_1}(t) = a\{1 - (1 + vt)\exp[-vt]\}. \tag{2.5}$$

ここで, $\{1 - (1 + vt)\exp[-vt]\}$ は, ソフトウェアシステム内におけるテスト空間占有率を表し, その占有率はS字形成長曲線に沿って拡大することを表す (図 2.4 参照). つまり, テスト開始時点において, テストケース設計者の潜在能力として発見可能なフォールト数が 0 個であり, テスト空間およびフォールト発見可能領域が存在しないことを意味する.

次に, テストケース設計者のテスト習熟性が高い場合について考えてみよう. この場合, 中堅社員などの熟練者によりテストケースが設計されることから, テスト

工程の初期段階でもテストケース設計者のフォールト発見可能性が高いため，設計されたテストケースを投入すればソフトウェアシステム内の広範囲のモジュールおよび機能に影響を及ぼす．つまり，テスト開始時点において，テストケース設計者の潜在能力として発見可能なフォールトが既に存在し，このようなテストケースを投入すれば，少数のテストケースでテスト空間がソフトウェア全体に拡大することを意味する．したがって，テストケース設計者のテスト習熟性が高い場合を考慮するために，微分方程式(2.3)および(2.4)を，それぞれ初期条件$w(0)=a(1-p)$および$u(0)=a(1-p)$の下で$u(t)$に関して解けばよいことがわかる．すなわち，次式を得る．

$$u(t)\equiv u_{b_2}(t)=a\{1-p(1+vt)\exp[-vt]\}\ (1\geq p\geq 0)\,. \tag{2.6}$$

ここで，pはテストケース設計者の非テスト習熟性を表すパラメータ，逆に$(1-p)$はテスト習熟性を意味すると同時に，テスト空間の初期拡大率を表す．この式(2.6)は，テスト空間占有率が図2.4のように拡大することを表す．また，式(2.6)において$p=1$のとき，$u_{b_2}(t)=u_{b_1}(t)$となり，テストケース設計者のテスト習熟性が低い場合を考慮した式(2.5)に帰着する．さらに$p=0$のとき，$u_{b_2}(t)=a$となりテスト開始直後においてテストケース設計者の潜在能力として発見可能なフォールト数が，テスト開始前にソフトウェアシステム内に潜在する総期待フォールト数となるようなテスト習熟性の極めて高い場合を表す．以上のことから，式(2.6)はテスト習熟性が低い場合と高い場合の両方を考慮した関数であるため，

$$u_{bS}(t)\equiv u_{b_2}(t)\,, \tag{2.7}$$

とし，$u_{bS}(t)$をテスト習熟性を考慮した簡易形テスト空間関数 (simplified testing-

domain function with skill-factor，以降**簡易習熟性 T-D 関数**と略す) と呼ぶことにする．

2.2.3 一般習熟性テスト空間関数

実際のテスト工程において，テストケース設計者のテスト習熟性が向上することにより発見可能となるフォールトの増加数，つまり単位テスト時間当りのフォールト発見可能領域の拡大率 v_1 と，テストケースを投入することにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体に潜在するフォールトの増加数，つまり単位テスト時間当りのテスト空間拡大率 v_2 とは，お互い独立した事象であるため，相異なる意味をもつパラメータと考える方が妥当であろう．

なぜなら，テストケース設計者は先ずテストするために，テスト対象となるソフトウェアシステムの要求仕様書および内部構造の理解に努め，それに応じたテストケースを設計する．このテストケースを設計する段階において，テスト習熟性は向上する．つまり，フォールト発見可能領域は拡大するが，テスト未実施であるためテスト空間は拡大しない．その後，テストのために設計されたテストケースが投入される．このテストケースの投入量およびその品質の度合に従ってテスト空間が形成される．このとき，テスト空間が拡大することにより新たに発見可能となるフォールトが認知される場合も考えられる．この場合，フォールト発見可能領域およびテスト空間は同時に拡大するものと考えられる．

つまり，テストケース設計者のテスト習熟性が向上することにより発見可能となるフォールトの増加現象を表す式 (2.3) のフォールト発見可能領域の拡大率 v_1 と，テ

スト空間内に潜在するフォールトの増加現象を表す式 (2.4) のテスト空間拡大率 v_2 の関係は次のように考えることができる：

$v_1 > v_2$：テスト習熟性の低いテストケース設計者がテストケースを設計しテストする場合，あるいは一般的なテスト習熟性を有するテストケース設計者が新規開発あるいは部品化されたモジュールの再利用率が低いソフトウェアシステムに対するテストケースを設計しテストする場合

$v_1 \approx v_2$：部品化されたモジュールの再利用率が高いソフトウェア製品を一般的なテスト習熟性を有するテストケース設計者がテストケースを設計しテストする場合

$v_1 < v_2$：テスト習熟性の高いテストケース設計者がテストケースを設計しテストする場合

以上のことから，簡易習熟性 T-D 関数の $u_{b_S}(t)$ のように，v_1 および v_2 が同一の値を示すことは考えにくい．また，v_1 および v_2 は，テスト空間を記述し，さらに SRGM を構築する上で重要なパラメータとなっているので，これらを厳密に捉えることは，SRGM の適合性および妥当性を高めるのに有効であると考えられる．

そこで，微分方程式 (2.3) および (2.4) の v_1 および v_2 は相異なるパラメータと考えて，簡易習熟性 T-D 関数と同様に，テストケース設計者のテスト習熟性の高い場合を表す初期条件 $w(0) = a(1-p)$ および $u(0) = a(1-p)$ の下で $u(t)$ に関して解くことにより次式を得る．

$$
\begin{aligned}
u(t) &\equiv u_{b_G}(t) \\
&= a\left\{1 + \frac{p\left(v_2 \cdot \exp\left[-v_1 t\right] - v_1 \cdot \exp\left[-v_2 t\right]\right)}{v_1 - v_2}\right\} \quad (v_1 \neq v_2). \qquad (2.8)
\end{aligned}
$$

ここで，式(2.8)に示す$u_{b_G}(t)$を，テスト習熟性を考慮した一般形テスト空間関数(generalized testing-domain function with skill-factor，以降**一般習熟性T-D関数**と略す)と呼ぶことにする．

特に，式(2.7)および(2.8)を合わせて，これらの関数のことを**習熟性T-D関数**と呼び，$u_b(t)$で表すことにする．

2.3 不完全デバッグ環境を考慮したテスト空間

本節では，ソフトウェア開発のテスト工程あるいは実際の運用・保守段階における，ソフトウェアシステム内に潜在するフォールトの発見および修正・除去，つまりデバッグ作業について考える．

これまでに構築されたSRGMの多くは，発見したソフトウェア故障に対して，その原因を分析し，ソフトウェア故障の原因であるフォールトを修正・除去するデバッグ作業がいつも確実に行われ，類似したソフトウェア故障は再発しないという**完全デバッグ(perfect debugging)環境**の仮定に基づいて構築されているが，この仮定は非現実的である．なぜなら，いつも確実にフォールト修正が行われるとは限らず，新規フォールトを作り込むことが実際には考え得るからである．ゆえに，ソフトウェア開発のテスト工程あるいは実際の運用・保守段階は，**不完全デバッグ(imperfect debugging)環境**にあるということを意味する[17],[18].

したがって，テスト開始前にソフトウェアシステム内に潜在する総期待フォールト数に対して，任意のテスト時刻tにおいてソフトウェアシステム内に潜在するフォールト数は増加することも考えられる．さらに，テスト工程後期あるいは運用・保守段階で発見されたフォールトを修正・除去する場合，テスト工程初期と比較して作

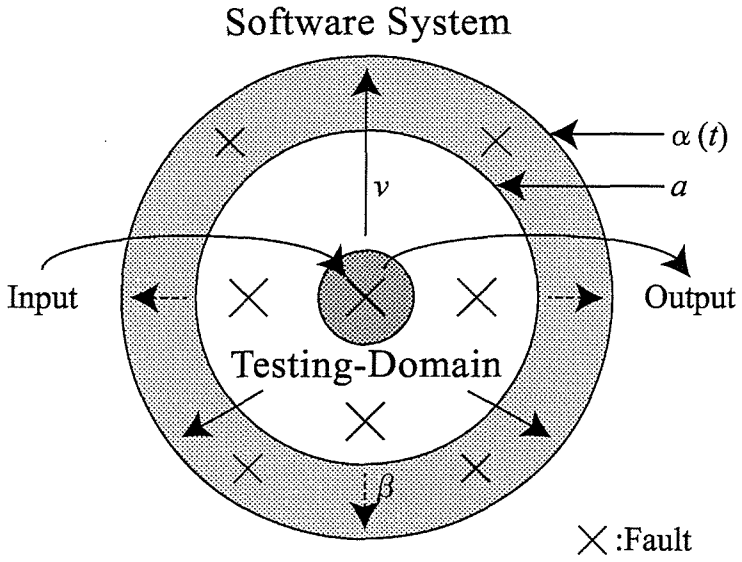

図 2.5: 不完全デバッグ環境を考慮したテスト空間概念図.

り込まれる新規フォールト数が，一定増分ではなく指数関数的に増大するものと推測でき，ソフトウェアシステムと共にテスト空間が拡大し続ける場合も考えられる(図2.5参照)．なぜなら，この時期に発見されるフォールトは，広範囲のモジュールおよび機能と複雑に絡み合い，修正により影響を及ぼす範囲が広大で，修正には細心の注意が必要となるからである．

次に，ソフトウェアシステム内におけるテスト空間占有率と，テストにより発見可能となるフォールト数との関係を定式化する．定式化するために，前述の議論より以下のような仮定を設定する [15]–[18].

1) フォールト修正時に新規フォールトが作り込まれることから，ソフトウェアシステム内に潜在する総累積フォールト数は増加する.

2) テスト空間内に潜在するフォールトは一様に分布する.

3) 任意のテスト時刻 t においてテスト空間内に潜在するフォールト数の増加率は，テスト空間外に残存する発見可能な総フォールト数に比例する．

これらの仮定から，本事象を次の微分方程式で表すことができる．

$$\frac{du(t)}{dt} = v[\alpha(t) - u(t)] \ (v > 0). \tag{2.9}$$

式 (2.9) において，$\alpha(t)$ は任意のテスト時刻 t においてソフトウェアシステム内に潜在する総累積フォールト数を表す．ここで，$\alpha(t)$ は以下に示す 2 つの式で表すことができる [18].

$$\alpha(t) \equiv \alpha_1(t) = a \cdot \exp[\beta t] \ (\beta > 0), \tag{2.10}$$

$$\alpha(t) \equiv \alpha_2(t) = a(1 + \epsilon t) \quad (\epsilon > 0). \tag{2.11}$$

ここで，β および ϵ は，フォールト修正時における新規フォールト潜入率を表す．本研究では，前述の議論より，テスト工程あるいは実際の運用・保守段階におけるデバッグ作業を忠実に表現する式 (2.10) を採用して，以下の議論を行う．

したがって，微分方程式 (2.9) に式 (2.10) を適用し，初期条件 $u(0) = 0$ の下で $u(t)$ に関して解くと次式を得る．

$$u(t) \equiv u_c(t) = \frac{av}{\beta + v}(\exp[\beta t] - \exp[-vt]). \tag{2.12}$$

式 (2.12) は，ソフトウェアシステム内におけるテスト空間占有率が図 2.6 のように拡大することを表し，$u_c(t)$ を不完全デバッグ環境を考慮したテスト空間関数 (testing-domain function with imperfect-debugging，以降**不完全デバッグ T-D 関数**と略す) と呼ぶことにする．また，式 (2.12) において $\beta = 0$ のとき，

$$u(t) = a(1 - \exp[-vt]), \tag{2.13}$$

となり，式(2.2)の$u_a(t)$で表される基本形T-D関数に帰着する[15],[16].

ここで，フォールト修正時における新規フォールト潜入率βは，

$\beta > 0$：テスト習熟性の低いデバッグ実施者によるテストおよびフォールト修正が行われている場合，あるいは一般的なテスト習熟性を有するデバッグ実施者により新規開発あるいは部品化されたモジュールの再利用率が低いソフトウェアシステムに対するテストおよびフォールト修正が行われている場合

$\beta \approx 0$: 高いテスト習熟性を有するデバッグ実施者によるテストおよびフォールト修正が行われている場合，あるいは一般的なテスト習熟性を有するデバッグ実施者により部品化されたモジュールの再利用率が高いソフトウェアシステムに対するテストおよびフォールト修正が行われている場合

と考えることができる.

2.4　テスト空間成長関数

本節では，前節までに導出されたそれぞれのテスト空間関数$u_a(t)$，$u_{b_S}(t)$，$u_{b_G}(t)$および$u_c(t)$に基づく**テスト空間成長関数(testing-domain growth function)**$\gamma(t)$を導く.

テスト空間成長関数は，単位テスト時間当りにテストケースを投入することにより拡大するテスト空間内において，発見可能となるフォールトの増加数，つまり発見可能となる瞬間フォールト増加率を表す．このテスト空間成長関数$\gamma(t)$は

$$\gamma(t) \equiv \frac{du(t)}{dt}, \tag{2.14}$$

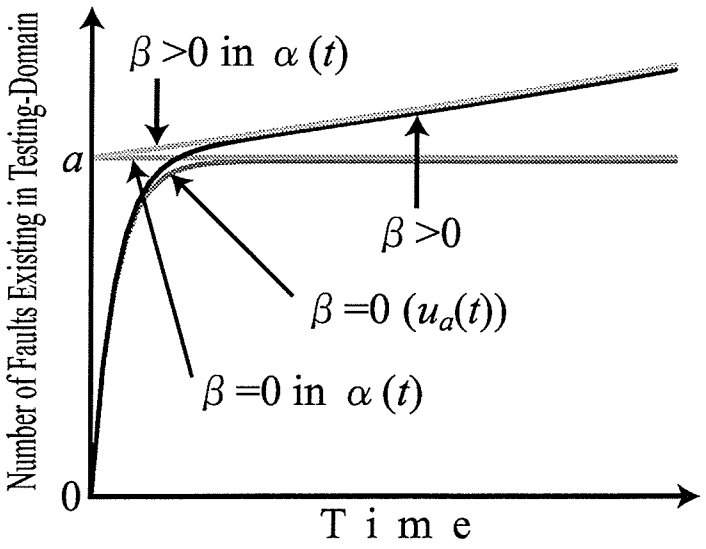

図 2.6: 不完全デバッグ T-D 関数 $u_c(t)$ の時間的推移.

により与えることができる．したがって，式 (2.14) において，テスト空間関数 $u(t)$ の代りに，式 (2.2) の基本形 T-D 関数 $u_a(t)$，式 (2.7) の簡易習熟性 T-D 関数 $u_{b_S}(t)$，式 (2.8) の一般習熟性 T-D 関数 $u_{b_G}(t)$，および式 (2.12) の不完全デバッグ T-D 関数 $u_c(t)$ を適用して，それぞれのテスト空間関数に基づくテスト空間成長関数は，

$$\gamma_a(t) = av \cdot \exp[-vt], \tag{2.15}$$

$$\gamma_{b_S}(t) = apv^2 t \cdot \exp[-vt], \tag{2.16}$$

$$\gamma_{b_G}(t) = \frac{apv_1v_2}{v_1 - v_2}\left(\exp[-v_2t] - \exp[-v_1t]\right), \tag{2.17}$$

$$\gamma_c(t) = \frac{av}{\beta + v}\left(\beta \cdot \exp[\beta t] + v \cdot \exp[-vt]\right), \tag{2.18}$$

のようにそれぞれ与えられる．

2.5　まとめ

本章では，テスト工程において，要求仕様書に基づいてインプリメントされた機能を検証するために投入されるテストケースにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体，つまりテスト空間について議論した．また，テスト空間のソフトウェアシステム内における占有率の時間的挙動に影響を及ぼすテスト要因，つまりテストケース設計者のテスト習熟性，およびフォールト修正時における新規フォールト混入可能性に着目したテスト空間についても議論した．

これらのテスト空間のソフトウェアシステム内における占有率の時間的推移がフォールト発見事象に影響するものと仮定したSRGMを構築することにより，テスト空間占有率の統計的推定値を導くことが可能になった．また，テスト管理者にとって重要な管理項目の１つである，テスト対象ソフトウェアシステム内のテストパス網羅率も擬似的に把握することができる．さらに，テスト空間成長関数$\gamma(t)$で表される，単位テスト時間当りにテストケースの投入により拡大するテスト空間内において発見可能となるフォールトの増加数が推定可能になった．これにより，テストケース設計者のテスト対象ソフトウェアシステムの要求仕様書および内部構造に対する理解度，つまりテスト習熟性の向上度合を定量的に表現することができる．

以上のことから，本章で提案した４種類のテスト空間に基づくSRGMを構築することは，テスト工程に対する妥当性とフォールトデータに対する適合性の改善，およびテスト管理者に対して重要な信頼性評価尺度の提供が可能となり，極めて有益であると考えられる．

第3章　ソフトウェア信頼度成長モデル

第２章では，ソフトウェア開発のテスト工程において，インプリメントした機能を検証するために投入されるテストケースにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体を意味するテスト空間について議論した．特に，テスト空間のソフトウェアシステム内における占有率の時間的挙動に多大な影響を及ぼすことが考えられるテスト要因を考慮した４種類のテスト空間について議論した．

本章では，発見されたフォールト数を計測することにより記述できる計数過程に対して，簡潔でかつ実際のフォールト発見事象の意味付けが容易であるNHPPに基づくSRGMについて議論する．SRGMは，高信頼ソフトウェア製品を開発し，その品質・信頼性管理に不可欠である定量的信頼性評価を実施するために，最も実用的で広範に適用されている数理モデルの１つである．そこで，3.1節ではNHPPの基礎概念について記述し，第２章で議論した４種類のテスト空間のソフトウェアシステム内における占有率の時間的推移が，フォールト発見事象に影響するものと仮定したSRGMを3.2節で提案する．さらに，提案する４種類のSRGMに基づく強度関数の導出も行う．3.3節では，提案する４種類のSRGMに含まれるモデルパラメータの推定方法として最ゆう法について議論する．3.4節では，SRGMから導出される定量的な信頼性評価尺度として，代表的な期待残存フォールト数，ソフトウェア

信頼度および平均ソフトウェア故障時間間隔について記述する．最後に，フォールトデータに対する適合性を統計的に検定するための方法として，コルモゴロフ・スミルノフ適合度検定法について3.5節で議論する．

3.1　非同次ポアソン過程

ある時間区間 $(0,\ t]$ において発生した事象の総数を変数 $N(t)$ で表すものとする．本研究では，ソフトウェア開発のテスト工程において発見されるフォールト数を $N(t)$ にあてはめて考える．このとき，$N(t)$ は確率的に変動する確率変数と考えることができ，しかも時間 t と共に変化するので確率変数の系，つまり確率過程 $\{N(t),\ t \geq 0\}$(stochastic process) として表される．特に，上記のような発生した事象数を表す確率過程を計数過程 (counting process) という．計数過程は，次のような性質をもつことが明らかである．

- $N(t) \geq 0$.
- $N(t)$ は整数値をとる．
- $t_1 \leq t_2$ ならば $N(t_1) \leq N(t_2)$ である．
- $t_1 \leq t_2$ ならば時間区間 $(t_1,\ t_2]$ で発生した事象数は $N(t_2) - N(t_1)$ である．

さらに，相異なる時間区間で発生する事象数が互いに統計的に独立であれば，計数過程 $\{N(t),\ t \geq 0\}$ は独立増分 (independent increments) 性をもつことを表す．例えば，$0 < t_1 < t_2$ のとき，$N(t_1)$ と $N(t_2) - N(t_1)$ は統計的に独立であることを意味する．

最も重要な計数過程の1つは**ポアソン過程 (Poisson process)** であり，特に次の性質を満足するとき，計数過程 $\{N(t),\ t \geq 0\}$ は NHPP であるという．

i. $N(0) = 0$.

ii. $\{N(t), t \geq 0\}$ は独立増分をもつ．

iii. $\Pr\{N(t+\Delta t) - N(t) \geq 2\} = o(\Delta t)$.

iv. $\Pr\{N(t+\Delta t) - N(t) = 1\} = r(t)\Delta t + o(\Delta t)$.

上記の特性 iv の $r(t)$ は NHPP の**強度関数 (intensity function)** と呼び，事象の発生率あるいは単位時間当りに発生する事象数を表す．また，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，

$$\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0,$$

と定義できる．さらに，性質 iv は，事象の発生しない確率が

$$\Pr\{N(t+\Delta t) - N(t) = 0\} = 1 - r(t)\Delta t + o(\Delta t),$$

であることを意味する．

NHPP の性質 i ～ iv を使って，確率変数 $N(t)$ の確率分布を求めると，

$$\Pr\{N(t) = n\} = \frac{\{A(t)\}^n}{n!} \exp[-A(t)] \ \ (n = 0, 1, 2, \cdots),$$

$$A(t) = \int_0^t r(x)\, dx, \tag{3.1}$$

を得る．式(3.1)において，$N(t)$の確率分布は平均値が$A(t)$の**ポアソン分布 (Poisson distribution)** に従うことを意味する．すなわち，

$$\mathrm{E}[N(t)] = \sum_{n=0}^{\infty} n \cdot \Pr\{N(t) = n\} = A(t), \tag{3.2}$$

である．また，式(3.1)の$A(t)$は確率変数$N(t)$の時間tに依存する平均値であるため，NHPPの**平均値関数 (mean value function)** と呼ぶ．したがって，$A(t)$は時間区間$(0,\ t]$において発生する総期待事象数を表す．NHPPの性質i～ivを使うと，$N(t)$の確率分布を

$$\Pr\{N(t+x) - N(t) = n\} = \frac{\{A(t+x) - A(t)\}^n}{n!} \exp[-\{A(t+x) - A(t)\}]$$

$$(t \geq 0,\ x \geq 0), \tag{3.3}$$

と表すこともでき，確率変数$N(t+x) - N(t)$が平均値$A(t+x) - A(t)$のポアソン分布に従うことがわかる．さらに，NHPPの性質ivにおいて，$r(t) = \lambda$(一定)として特別な場合を考えれば，すなわち時刻tに関係なく事象の発生率が一定のときは，平均値関数は$A(t) = \lambda t$となるため，式(3.1)は

$$\Pr\{N(t) = n\} = \frac{(\lambda t)^n}{n!} \exp[-\lambda t] \ \ (n = 0, 1, 2, \cdots), \tag{3.4}$$

となり，計数過程$\{N(t),\ t \geq 0\}$は同次ポアソン過程(homogeneous Poisson process)であるという．このとき，式(3.3)から$\{N(t+x) - N(t)\}$は平均値がλxのポアソン分布に従うことがわかり，同次ポアソン過程のときには過去の履歴とは独立であることになる．

3.2 NHPPに基づくモデル化

本研究では，NHPPに基づくSRGMを取り上げて議論している．ソフトウェア開発の最終工程であるテスト段階におけるフォールト発見事象をモデル化するために，任意のテスト時刻$t\,(t \geq 0)$までに発見されるフォールトの累積数を$N(t)$とすると，$N(t)$は時刻tまでに発見された総累積フォールト数を表す計数過程$\{N(t),\ t \geq 0\}$とみなすことができる．この計数過程，つまりテスト工程におけるフォールト発見事象に対して，前節で議論したNHPPを次のように仮定する．

i. $N(0) = 0$:

テスト時刻$t = 0$においてフォールト発見数は0件である．

ii. $\{N(t), t \geq 0\}$は独立増分をもつ：

異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．

iii. $\Pr\{N(t + \Delta t) - N(t) \geq 2\} = o(\Delta t)$:

任意の微小テスト時間区間Δtで2個以上のフォールトが発見される確率は無視できるほど小さい．

iv. $\Pr\{N(t + \Delta t) - N(t) = 1\} = h(t)\,\Delta t + o(\Delta t)$:

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．

このとき，フォールト発見事象は，NHPPにより

$$\Pr\{N(t)=n\}=\frac{\{H(t)\}^n}{n!}\exp[-H(t)]\ (n=0,1,2,\cdots),$$

$$H(t)=\int_0^t h(x)\,dx, \tag{3.5}$$

と定式化される．式(3.5)において，$\Pr\{X\}$は事象Xが発生し得る確率を表す．また，$H(t)$はNHPPの平均値関数であり，時刻tまでに発見される総累積フォールト数$N(t)$の期待値を表す．さらに，$h(t)$はNHPPの強度関数であり，単位テスト時間当りに発見されるフォールト数，つまり瞬間フォールト発見率を表す．

上述の議論より，ソフトウェアシステム内におけるテスト空間占有率の時間的推移がフォールト発見事象に影響するものと仮定したSRGMを記述するために，次のような仮定を設定する．

1) 発見されるフォールトは，必ずテスト空間内に存在する．

2) ソフトウェアシステム内におけるテスト空間占有率は，テスト時間の経過と共に増大する．

3) 単位テスト時間当りに発見されるフォールト数は，テスト空間内に残存するフォールト数に比例する．

上記の仮定3)およびテスト空間関数$u(t)$から，平均値関数$H(t)$に関して次の微分方程式が成立する．

$$\frac{dH(t)}{dt}=b[u(t)-H(t)]\ (1>b>0). \tag{3.6}$$

式(3.6)において，bは比例定数であり，テスト空間内に残存するフォールト1個当りの発見率を表す．

式 (3.6) で表される微分方程式において，テスト空間関数 $u(t)$ の代りに，式 (2.2) の基本形 T-D 関数 $u_a(t)$，式 (2.7) の簡易習熟性 T-D 関数 $u_{b_S}(t)$，式 (2.8) の一般習熟性 T-D 関数 $u_{b_G}(t)$，および式 (2.12) の不完全デバッグ T-D 関数 $u_c(t)$ を適用する．さらに，NHPP の性質 i ($N(0)=0$) から初期条件 $H(0)=0$ の下で平均値関数 $H(t)$ に関して解くことにより，それぞれ次式を得る．

$$H_a(t) = a\left\{1 + \frac{b\cdot\exp[-vt] - v\cdot\exp[-bt]}{v-b}\right\} \qquad (v \neq b), \tag{3.7}$$

$$H_{b_S}(t) = a\left\{1 + \frac{bp}{v-b}\left(vt + \frac{2v-b}{v-b}\right)\exp[-vt] - \left[1 + \frac{bp(2v-b)}{(v-b)^2}\right]\exp[-bt]\right\} \quad (v \neq b), \tag{3.8}$$

$$H_{b_G}(t) = a\left\{1 - \frac{bpv_2\cdot\exp[-v_1t]}{(v_1-v_2)(v_1-b)} + \frac{bpv_1\cdot\exp[-v_2t]}{(v_1-v_2)(v_2-b)} - \left[1 - \frac{bp(b-v_1-v_2)}{(v_1-b)(v_2-b)}\right]\exp[-bt]\right\} \quad (v_1 \neq v_2 \neq b), \tag{3.9}$$

$$H_c(t) = abv\left\{\frac{\exp[\beta t]}{(\beta+v)(\beta+b)} + \frac{\exp[-vt]}{(\beta+v)(v-b)} - \frac{\exp[-bt]}{(\beta+b)(v-b)}\right\} \quad (v \neq b). \tag{3.10}$$

ここで，それぞれ導出された平均値関数について

- $H_a(t)$ を基本形 T-D 関数 $u_a(t)$ に基づく基本形 T-D SRGM
- $H_{b_S}(t)$ を簡易習熟性 T-D 関数 $u_{b_S}(t)$ に基づく簡易習熟性 T-D SRGM

・ $H_{b_G}(t)$ を一般習熟性 T-D 関数 $u_{b_G}(t)$ に基づく一般習熟性 T-D SRGM

・ $H_c(t)$ を不完全デバッグ T-D 関数 $u_c(t)$ に基づく不完全デバッグ T-D SRGM

と呼ぶことにする.

次に，導出されたそれぞれの平均値関数について考察する.

先ず，式 (3.10) において，$\beta = 0$ のとき，フォールト修正時に新規フォールトの作り込みが無いことを表す場合に対応し，平均値関数 $H_a(t)$ で表される式 (3.7) の基本形 T-D SRGM に帰着する．この条件の下で，式 (3.7) および (3.10) において，$v \to \infty$ のとき，これはテスト開始直後にテスト空間がソフトウェアシステム全体に拡大する場合に対応し，テスト空間関数はテスト開始前にソフトウェアシステム内に潜在する総期待フォールト数を表す

$$u_a(t) = u_c(t) = a,$$

となり，既存の指数形 SRGM に帰着する [12]．同様に $\beta = 0$ の条件下で，$v = b$ のとき，これはテスト空間が拡大すると同時にフォールトが発見される場合に対応し，テスト空間関数はソフトウェアシステム内におけるテスト空間占有率が指数形成長曲線を表す

$$u_a(t) = u_c(t) = a\{1 - \exp[-bt]\},$$

となり，既存の遅延 S 字形 SRGM に帰着する [13].

また，式 (3.8) および (3.9) において，$p = 0$ のとき，これはテストケース設計者のテスト習熟性が最も高い場合に対応し，テスト空間関数はテスト開始直後にソフトウェアシステム全体に拡大したことを表す

$$u_{b_S}(t) = u_{b_G}(t) = a,$$

となり，既存の指数形 SRGM に帰着する [12].

以上のことから，第２章で議論した４種類のテスト空間関数に基づく SRGM は，指数形 SRGM[12] および遅延Ｓ字形 SRGM[13] の両特徴を兼ね備えた SRGM であることが伺える．ゆえに，これらのテスト空間依存型 SRGM は，指数形成長曲線およびＳ字形成長曲線を示す両タイプのフォールトデータに適用できる柔軟性の高いモデルであるといえる．

さらに，単位テスト時間当りに発見されるフォールト数，つまり式 (3.5) における瞬間フォールト発見率を表す NHPP の強度関数 $h(t)$ は，式 (3.7)-(3.10) から，

$$h_a(t) = \frac{abv}{v-b}\left\{\exp[-bt] - \exp[-vt]\right\} \qquad (v \neq b), \tag{3.11}$$

$$h_{b_S}(t) = ab\left\{\left[1 + \frac{bp(2v-b)}{(v-b)^2}\right]\exp[-bt] - \frac{pv^2}{v-b}\left(t + \frac{1}{v-b}\right)\exp[-vt]\right\} \quad (v \neq b), \tag{3.12}$$

$$h_{b_G}(t) = ab\left\{\frac{pv_1v_2}{v_1-v_2}\left(\frac{\exp[-v_1t]}{v_1-b} - \frac{\exp[-v_2t]}{v_2-b}\right) + \left[1 - \frac{bp(b-v_1-v_2)}{(v_1-b)(v_2-b)}\right]\exp[-bt]\right\} \quad (v_1 \neq v_2 \neq b), \tag{3.13}$$

$$h_c(t) = abv\left\{\frac{\beta\cdot\exp[\beta t]}{(\beta+v)(\beta+b)} - \frac{v\cdot\exp[-vt]}{(\beta+v)(v-b)} + \frac{b\cdot\exp[-bt]}{(\beta+b)(v-b)}\right\} \qquad (v \neq b), \tag{3.14}$$

のようにそれぞれ与えられる．

3.3　パラメータの推定方法

SRGMにより実測フォールトデータを解析するために，3.2節で議論した4種類の平均値関数 $H(t)$，つまり式(3.7)の基本形T-D SRGM $H_a(t)$，式(3.8)の簡易習熟性T-D SRGM $H_{b_S}(t)$，式(3.9)の一般習熟性T-D SRGM $H_{b_G}(t)$，および式(3.10)の不完全デバッグT-D SRGM $H_c(t)$ に含まれる信頼度成長パラメータ a，b，v，v_1，v_2，p および β を推定する必要がある．そこで，本節では実測フォールトデータが確率的に最も実現されやすいように信頼度成長パラメータを推定する方法である**最ゆう法 (method of maximum-likelihood)**[3]–[6] について議論する．

ソフトウェア開発のテスト工程において採取されたフォールトデータとして，一定の時間区間 $(0,\ t_k]$ において発見された総累積フォールト数 y_k に関するデータが n 組観測されたものとする $(t_k,\ y_k)\,(k=1,2,\cdots,n;0<t_1<t_2<\cdots<t_n)$．ここで，$n$ 組の測定データの組を $\mathbf{t}=(t_1,t_2,\cdots,t_n)$ および $\mathbf{y}=(y_1,y_2,\cdots,y_n)$ により表す．実測フォールトデータ $\mathbf{t}$ および $\mathbf{y}$ が与えられたときの，平均値関数 $H(t)$ をもつNHPPモデルの信頼度成長パラメータに対する**ゆう度関数 (likelihood function)** は，$\{N(t_1)=y_1,\ N(t_2)=y_2,\ \cdots,\ N(t_n)=y_n\}$ の同時確率変数であるから，NHPPの性質ii (独立増分) から

$$
\begin{aligned}
L &\equiv \{N(t_1)=y_1, N(t_2)=y_2,\cdots,N(t_n)=y_n\} \\
&= \prod_{k=1}^{n} \Pr\{N(t_k)-N(t_{k-1})=y_k-y_{k-1}\} \\
&= \prod_{k=1}^{n} \frac{\{H(t_k)-H(t_{k-1})\}^{(y_k-y_{k-1})}}{(y_k-y_{k-1})!}\cdot\exp[-H(t_n)], \qquad (3.15)
\end{aligned}
$$

となる．ここで，$t_0 \equiv 0$ および $y_0 \equiv 0$ とする．式 (3.15) の両辺の自然対数をとると，

$$\ln L = \sum_{k=1}^{n} (y_k - y_{k-1}) \cdot \ln [H(t_k) - H(t_{k-1})]$$

$$- \sum_{k=1}^{n} \ln [(y_k - y_{k-1})!] - H(t_n), \tag{3.16}$$

となる．式 (3.16) において，それぞれの平均値関数 $H(t)$ に含まれる各パラメータについて偏微分し，

$$\frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial b} = \frac{\partial \ln L}{\partial v} = \frac{\partial \ln L}{\partial v_1} = \frac{\partial \ln L}{\partial v_2} = \frac{\partial \ln L}{\partial p} = \frac{\partial \ln L}{\partial \beta} = 0, \tag{3.17}$$

となる同時ゆう度方程式を数値的に解くことにより，**最ゆう推定値 (maximum likelihood estimate)** $\widehat{a}$，$\widehat{b}$，$\widehat{v}$，$\widehat{v}_1$，$\widehat{v}_2$，$\widehat{p}$ および $\widehat{\beta}$ が得られる．

3.4 信頼性評価尺度

NHPP により記述される SRGM に基づいて，様々な定量的信頼性評価尺度が研究されている．ここでは，代表的な信頼性評価尺度 [3]–[6] である

1) **期待残存フォールト数** (expected number of remaining faults)

2) **ソフトウェア信頼度** (software reliability)

3) **平均ソフトウェア故障時間間隔**

(mean time between software failures，以降 MTBF と略す)

について議論する．

3.4.1　期待残存フォールト数

テスト時刻 t におけるソフトウェアシステム内の期待残存フォールト数について考える (図 3.1 参照).

この尺度において，任意のテスト時刻 t におけるソフトウェアシステム内の残存フォールト数を $\overline{N}(t)$ により表すと，

$$\overline{N}(t) = N(\infty) - N(t),$$

と定義でき，$\overline{N}(t)$ の平均値 $n(t)$ を求めると，$H(\infty) = a$ から

$$\begin{aligned} n(t) \equiv \mathrm{E}\left[\overline{N}(t)\right] &= a - H(t) \\ &= \mathrm{Var}\left[\overline{N}(t)\right], \end{aligned} \tag{3.18}$$

となり，$\overline{N}(t)$ の分散と同一であることがわかる.

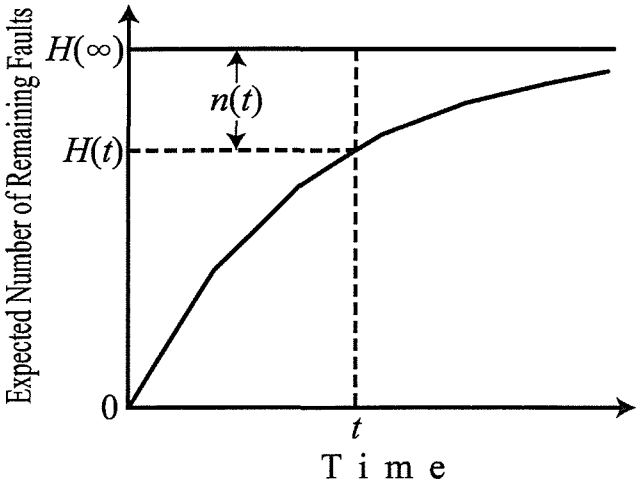

図 3.1: 期待残存フォールト数.

3.4.2 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 $(t,\ t+x]$ $(t \geq 0,\ x \geq 0)$ においてソフトウェア故障の発生しない確率について考える．

ソフトウェアシステムが時刻 $t=0$ で動作を開始したときに，k 番目のソフトウェア故障発生時刻を表す確率変数 S_k $(k=1,2,\cdots)$ を定義する．このとき，

$$X_k = S_k - S_{k-1}\ (k=1,2,\cdots\ ;\ S_0=0),$$

は，$(k-1)$ 番目と k 番目の故障間の時間間隔を表す確率変数となる (図 3.2 参照)．そこで，ソフトウェア故障発生現象が NHPP により記述されるとき，$(k-1)$ 番目の故障が時刻 t で発生したときに，次の k 番目の故障が時間区間 $(t,\ t+x]$ で発生しない条件付き確率を考える (図 3.3 参照)．これは，NHPP の性質 ii (独立増分) を使うと，

$$\begin{aligned} R(x|t) &\equiv \Pr\{X_k > x | S_{k-1} = t\} \\ &= \Pr\{N(t+x) - N(t) = 0\} \\ &= \exp[-\{H(t+x) - H(t)\}], \end{aligned} \tag{3.19}$$

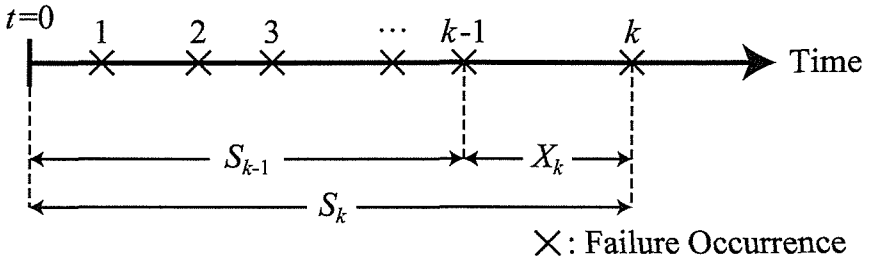

図 3.2: ソフトウェア故障発生時間の記述．

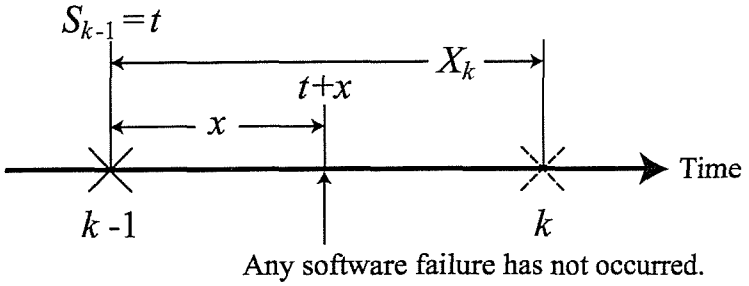

図 3.3: ソフトウェア信頼度の概念図.

となる．ただし，$H(\infty)=a$ であることに注意する．これは，テスト時刻 t でソフトウェア故障が起こったという条件の下で，時間区間 $(t,\ t+x]$ でソフトウェア故障の発生しない確率を意味する．

3.4.3 MTBF

前節とは逆に時間区間 $(t,\ t+x]$ $(t\geq 0,\ x\geq 0)$ においてソフトウェア故障が発生する確率について考える．

この尺度は，式 (3.19) から

$$\begin{aligned} F(x|t) &= 1-R(x|t) \\ &= 1-\exp\left[-\{H(t+x)-H(t)\}\right], \end{aligned} \tag{3.20}$$

により与えられるが，

$$F(0|t)=0, \qquad F(\infty|t)=1-\exp\left[-n(t)\right],$$

となるので，通常の分布関数の条件 $(F(\infty|t)=1)$ を満足していないことがわかる．したがって，ソフトウェア故障発生時間間隔の平均値，すなわちMTBFを導出でき

ない．しかし，MTBF の代替的尺度として，NHPP の強度関数の逆数，つまり瞬間的な平均ソフトウェア故障時間間隔を表す

$$MTBF_{\mathrm{I}}(t) = \frac{1}{h(t)}, \tag{3.21}$$

および，テスト時間を NHPP の平均値関数で割った値，つまり瞬間的な平均フォールト発見時間間隔を表す

$$MTBF_{\mathrm{C}}(t) = \frac{t}{H(t)}, \tag{3.22}$$

を定義することができる．式 (3.21) および (3.22) は，それぞれテスト時刻 t における**瞬間 MTBF**(instantaneous MTBF)，**累積 MTBF**(cumulative MTBF) と呼ばれている [3]–[6].

3.5 適合度検定

3.2 節で導出した平均値関数 $H(t)$ をもつ NHPP モデルの実測フォールトデータに対する統計的な適合性を確認する必要がある．そこで，本節では実測フォールトデータ数が少ない場合でも効果的な統計的検定法として知られる**コルモゴロフ・スミルノフ適合度検定法** (Kolmogorov-Smirnov goodness-of-fit test，以降 K–S 検定法と略す)[3]–[6],[19] について議論する．

検定される仮説により決まる連続的確率変数 X の実測値 n 個を小さい順に並び替えた統計量，すなわち順序統計量とする．このとき，コルモゴロフ・スミルノフ検定統計量 (以降，K–S 統計量と略す) は，

$$D = \max_{1 \le i \le n} \{D_i\},$$

$$D_i = \max\left\{\left|F(x_i) - \frac{i}{n}\right|, \left|F(x_i) - \frac{i-1}{n}\right|\right\}, \tag{3.23}$$

となる．平均値関数 $H(t)$ をもつNHPPモデルでは，

$$F(x_i) = \frac{H(x_i)}{H(x_n)}, \tag{3.24}$$

である．したがって，フォールト発見数データ $(t_k,\ y_k)\,(k = 1, 2, \cdots, n\ ;\ 0 < t_1 < t_2 < \cdots < t_n)$ が観測されたときには，

$$\begin{aligned} D &= \max_{1 \le i \le n}\{D_i\}, \\ D_i &= \max\left\{\left|\frac{H(t_i)}{H(t_n)} - \frac{y_i}{y_n}\right|, \left|\frac{H(t_i)}{H(t_n)} - \frac{y_{i-1}}{y_n}\right|\right\}, \end{aligned} \tag{3.25}$$

となるK–S統計量を計算する．K–S検定法では，データから計算されたK–S統計量 D と有意水準が θ (1%あるいは5%) のときの棄却限界と比較される．式(3.25)において，自由度が n のときの棄却限界 $D_{n\ ;\ \theta}$ と比較し，

$$D < D_{n\ ;\ \theta}\ ,$$

ならば，有意水準 θ で実測フォールトデータに対して，平均値関数 $H(t)$ をもつNHPPモデルは適合していると判定できる．一方，

$$D > D_{n\ ;\ \theta}\ ,$$

となってNHPPモデルが適合しなければ，追加データを収集して再びNHPPモデルを推定し直すか，他のNHPPモデルを適用する (なお，棄却限界 $D_{n\ ;\ \theta}$ については統計数値表 [20] を参照のこと)．

3.6 まとめ

本章では，フォールト発見事象を記述するためのNHPPについて説明し，第2章で議論した4種類のテスト空間のソフトウェアシステム内における占有率の時間的挙動と関係づけたNHPPに基づくSRGMを構築した．さらに，構築した4種類のSRGMの各モデルパラメータの推定方法として，実測フォールトデータが確率的に最も実現され易いように推定する最ゆう法，および推定結果が実測フォールトデータに対する適合性を統計的に検定するための方法であるK–S検定法について議論した．また，これらのSRGMから導出される定量的信頼性評価尺度，つまり瞬間フォールト発見率，期待残存フォールト数，ソフトウェア信頼度，瞬間MTBF，および累積MTBFについても議論した．

したがって，実際のフォールト発見事象を的確に表現し，テスト工程に対する適合性を向上させるために着目したテスト要因を考慮したテスト空間依存型SRGMから導出されるこれらの信頼性評価尺度は，既存のSRGMよりも高精度で予測することが可能であると考えられる．ゆえに，これらの信頼性評価尺度を併用したテスト管理を実施することで，効率的かつ経済的に高品質・信頼性ソフトウェアシステムの開発が可能になると思われる．

第4章　実測データに対するモデルの適用と評価

第３章では，ソフトウェアシステム内におけるテスト空間占有率の時間的変化に影響を及ぼすことが考えられるテスト要因を考慮した４種類のテスト空間に基づくSRGMについて議論した．また，導出したSRGMのモデルパラメータの推定方法，およびSRGMから導出される信頼性評価尺度，フォールトデータに対する適合性を統計的に検定するためのK–S検定法についても議論した．

本章では，第３章で議論したSRGMに実測フォールトデータを適用した結果について考察を行う．ここで用いた実測フォールトデータは４種類で，いずれも実際のソフトウェア開発現場で観測されたものである．そこで，4.1節において，3.3節で議論した最ゆう法を用いて，４種類のテスト空間依存型SRGMの各モデルパラメータの推定を行い，テスト空間占有率の時間的変化に影響を及ぼすと考えられるパラメータについて4.2節で考察する．また，4.3節ではテスト空間依存型SRGMから導出される定量的な信頼性評価尺度の推定結果の考察を行い，今後実施し得るテスト施策について議論する．さらに，4.4節では３種類の適合性評価基準を用いた比較結果を示し，4.5節でその比較結果を考察する．

4.1　モデルパラメータの推定結果

第3章で議論した4種類のテスト空間依存型SRGMの妥当性を評価するために，実際のソフトウェア開発のテスト工程において観測されたフォールトデータにこれらのモデルを適用する．適用するデータセットは，ソフトウェア開発プロジェクトのテスト工程の現場において採取されたフォールトデータであり，そのテスト時間に対する累積フォールト数の傾向が，S字形成長曲線としてプロットされるデータセットをDS1およびDS2，および指数形成長曲線としてプロットされるデータセットをDS3およびDS4として，以下のように表すことにする：

DS1：$(t_k,\ y_k)\,(k=1,2,\cdots,44\ ;\ t_k(\text{days}))$, 47.6×10^3 LOC,

DS2：$(t_k,\ y_k)\,(k=1,2,\cdots,24\ ;\ t_k(\text{days}))$, 26.0×10^3 LOC,

DS3：$(t_k,\ y_k)\,(k=1,2,\cdots,42\ ;\ t_k(\text{days}))$, 41.7×10^3 LOC,

DS4：$(t_k,\ y_k)\,(k=1,2,\cdots,40\ ;\ t_k(\text{days}))$, 31.5×10^3 LOC.

ここで，テスト時刻t_kの計測単位はカレンダー日を表す．また，LOC(line of codes)はコード行数を表す．さらに，開発言語はDS1～DS4共にC言語，C++言語およびアセンブラである．フォールトデータの採取環境は，それぞれソフトウェアシステムの要求仕様書に基づいて予め設計したテストケースを順次テストしていき，その際に観測されたフォールト数をカレンダー日単位で計測したものである．

このDS1～DS4を用いて，3.3節で議論した最ゆう法により，式(3.17)で表される同時ゆう度方程式を，それぞれ数値的に解くことにより信頼度成長パラメータの最ゆう推定値を求めた．DS1～DS4に対する平均値関数$H_a(t)$の推定結果を表4.1，$H_{b_S}(t)$を表4.2，$H_{b_G}(t)$を表4.3，および$H_c(t)$を表4.4に示す．

表 4.1: $H_a(t)$ のモデルパラメータの推定結果.

Data Set	$\widehat{a}$	$\widehat{b}$	$\widehat{v}$
DS1	164.350	0.057412	0.057403
DS2	126.604	0.089990	0.528041
DS3	50.467	0.416222	0.119563
DS4	64.222	0.056054	1.011859

表 4.2: $H_{b_S}(t)$ のモデルパラメータの推定結果.

Data Set	$\widehat{a}$	$\widehat{b}$	$\widehat{v}$	$\widehat{p}$
DS1	152.301	0.135613	0.076572	0.910179
DS2	130.402	0.080051	1.443637	1.000000
DS3	51.038	0.096963	0.958072	0.815407
DS4	62.121	0.073143	0.185153	0.324055

表 4.3: $H_{b_G}(t)$ のモデルパラメータの推定結果.

Data Set	$\widehat{a}$	$\widehat{b}$	$\widehat{v_1}$	$\widehat{v_2}$	$\widehat{p}$
DS1	152.282	0.135448	0.076613	0.076611	0.910098
DS2	127.288	0.087777	5.204254	0.632317	0.999869
DS3	51.039	0.096952	0.961454	0.955550	0.815501
DS4	64.206	0.056087	2.019751	1.979356	0.988577

なお，平均値関数 $H(t)$，つまり式 (3.7) の $H_a(t)$，式 (3.8) の $H_{b_S}(t)$，式 (3.9) の $H_{b_G}(t)$，および式 (3.10) の $H_c(t)$ をもつNHPPモデルの実測フォールトデータに対する統計的な適合性を調べるために，3.5節で議論したK–S検定を実施した結果を

表 4.4: $H_c(t)$ のモデルパラメータの推定結果.

Data Set	$\widehat{a}$	$\widehat{b}$	$\widehat{v}$	$\widehat{\beta}$
DS1	164.424	0.054146	0.060968	1.59×10^{-7}
DS2	116.844	0.443461	0.106486	2.57×10^{-3}
DS3	42.870	0.136448	0.484724	4.81×10^{-3}
DS4	64.223	0.056053	1.011896	2.50×10^{-8}

以下に示す.

平均値関数 $H_a(t)$ の場合

$$\mathrm{DS1}: \mathrm{D}_{44;0.05} = 0.15796 > 0.10599, \qquad \mathrm{DS2}: \mathrm{D}_{24;0.05} = 0.21204 > 0.12622,$$

$$\mathrm{DS3}: \mathrm{D}_{42;0.05} = 0.16158 > 0.11170, \qquad \mathrm{DS4}: \mathrm{D}_{40;0.05} = 0.16547 > 0.07919.$$

平均値関数 $H_{b_S}(t)$ の場合

$$\mathrm{DS1}: \mathrm{D}_{44;0.05} = 0.15796 > 0.09581, \qquad \mathrm{DS2}: \mathrm{D}_{24;0.05} = 0.21204 > 0.13763,$$

$$\mathrm{DS3}: \mathrm{D}_{42;0.05} = 0.16158 > 0.10194, \qquad \mathrm{DS4}: \mathrm{D}_{40;0.05} = 0.16547 > 0.09538.$$

平均値関数 $H_{b_G}(t)$ の場合

$$\mathrm{DS1}: \mathrm{D}_{44;0.05} = 0.15796 > 0.09582, \qquad \mathrm{DS2}: \mathrm{D}_{24;0.05} = 0.21204 > 0.12869,$$

$$\mathrm{DS3}: \mathrm{D}_{42;0.05} = 0.16158 > 0.10195, \qquad \mathrm{DS4}: \mathrm{D}_{40;0.05} = 0.16547 > 0.07945.$$

平均値関数 $H_c(t)$ の場合

$$\mathrm{DS1}: \mathrm{D}_{44;0.05} = 0.15796 > 0.10628, \qquad \mathrm{DS2}: \mathrm{D}_{24;0.05} = 0.21204 > 0.12391,$$

$$\mathrm{DS3}: \mathrm{D}_{42;0.05} = 0.16158 > 0.11388, \qquad \mathrm{DS4}: \mathrm{D}_{40;0.05} = 0.16547 > 0.07919.$$

上記に示すK–S検定結果より，提案した4種類のテスト空間依存型SRGMは，DS1～DS4に対して有意水準5% で十分に適合していることがわかった．

さらに，提案した4種類のテスト空間依存型SRGMにDS1～DS4を適用して得られた信頼度成長パラメータの推定値を用いて，式(3.7)の平均値関数$H_a(t)$を，発見された総累積フォールト数(DS1) および90% 信頼限界と共に図4.1に示す．また，式(3.8)の$H_{b_S}(t)$およびDS2，式(3.9)の$H_{b_G}(t)$およびDS3，および式(3.10)の$H_c(t)$およびDS4を，図4.2～図4.4にそれぞれ示す．

ここで，有意水準θで推定される平均値関数$\widehat{H}(t)$の存在範囲，すなわち100θ%信頼限界を，

$$\widehat{H}(t) \pm K_\theta \sqrt{\widehat{H}(t)} \quad ,$$

により計算する．K_θは標準正規分布の$100(1+\theta)/2$パーセント点である．

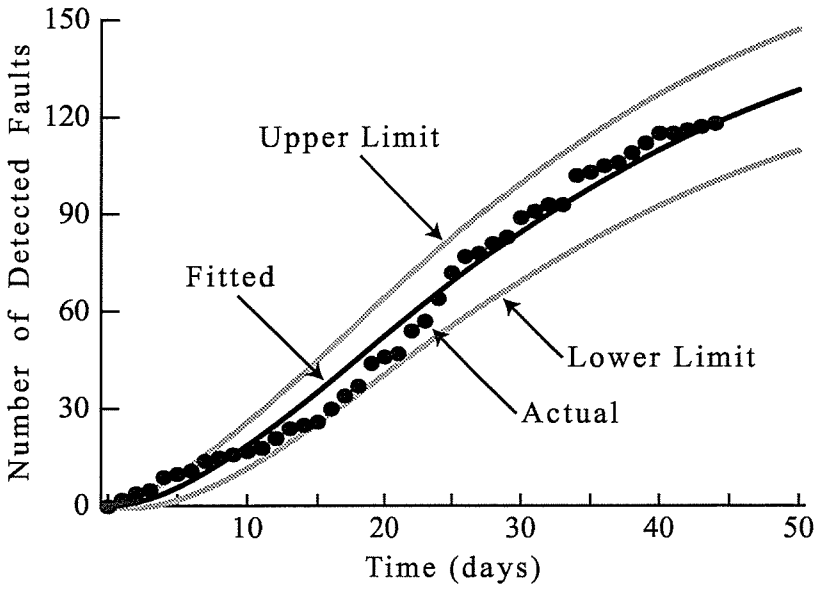

図4.1: 推定された平均値関数$H_a(t)$ (DS1).

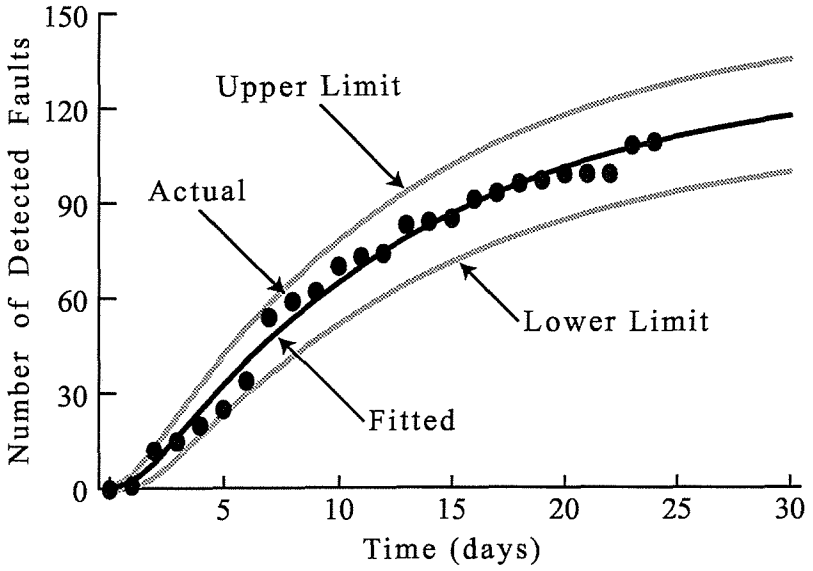

図 4.2: 推定された平均値関数 $H_{b_S}(t)$ (DS2).

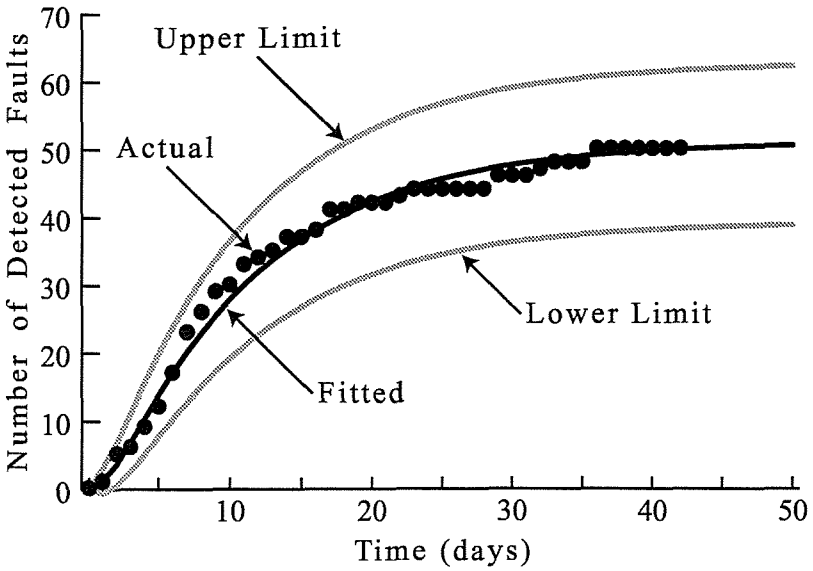

図 4.3: 推定された平均値関数 $H_{b_G}(t)$ (DS3).

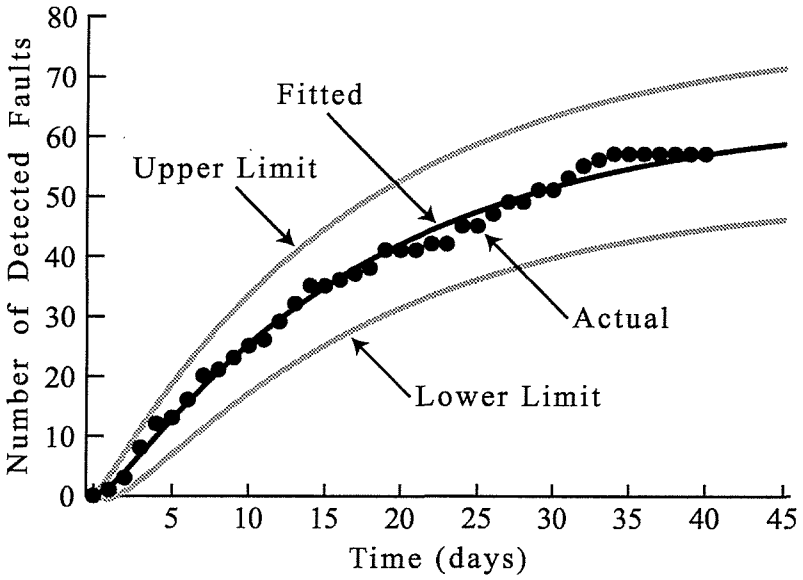

図 4.4: 推定された平均値関数 $H_c(t)$ (DS4).

4.2 推定結果の考察

次に，ソフトウェアシステム内におけるテスト空間占有率の時間的推移に影響を及ぼすテスト要因を反映したモデルパラメータの推定結果について考察する．つまり，テストケース設計者のテスト習熟性を考慮したモデル，特に一般習熟性 T-D SRGM においては，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 の関係，およびテスト習熟性 p から，ソフトウェアシステム内の部品化されたモジュールの再利用率，およびテストケース設計者のテスト習熟性について考察すると共に，簡易習熟性 T-D SRGM および一般習熟性 T-D SRGM の関係についても考察する．また，フォールト修正時における新規フォールト混入可能性を考慮した不完全デバッグ T-D SRGM においても，テスト空間拡大率 v および新規フォールト潜入率 β から，同様の考察を行う．

4.2.1　習熟性 T-D 関数に基づく SRGM

DS1 について

DS1 に対する式 (2.7) の簡易習熟性 T-D 関数，および式 (2.8) の一般習熟性 T-D 関数で表されるテスト空間占有率の時間的変化を，それぞれ図 4.5 および図 4.6 に示す．DS1 については，一般習熟性 T-D SRGM の推定値が $v_1 \approx v_2$ であることから，これは部品化されたモジュールの再利用率が高いソフトウェア製品を一般的なテスト習熟性を有するテストケース設計者によりテストケースが設計され，テストされている場合であることが推測できる．これは，パラメータ p の推定値からも同様のことが伺える．さらに，表 4.2 および表 4.3 の推定結果から，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 の関係が $v_1 \approx v_2$ であること，および

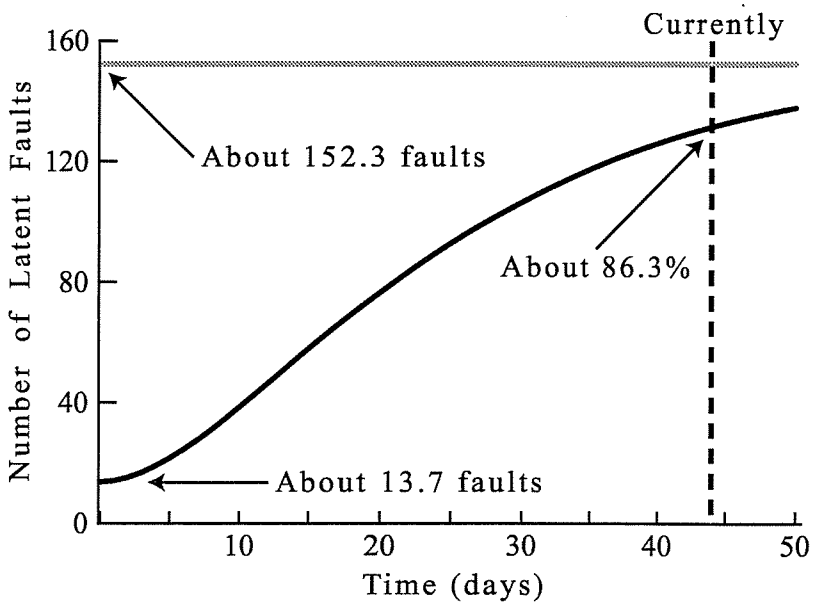

図 4.5: 推定された簡易習熟性 T-D 関数 $u_{b_S}(t)$ (DS1).

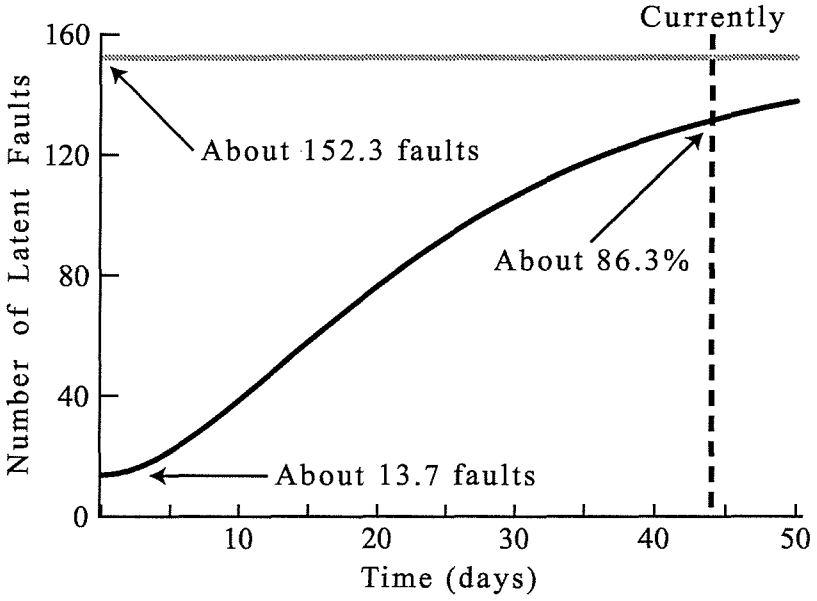

図 4.6: 推定された一般習熟性 T-D 関数 $u_{b_G}(t)$ (DS1).

両 SRGM のモデルパラメータの推定値がほぼ同一であることがわかる．また，テスト空間占有率の時間的推移を表す図 4.5 および図 4.6 においても同一の結果が得られている．以上のことから，本データセットの場合，モデルパラメータの少ない簡易習熟性 T-D SRGM を用いた定量的信頼性評価において，近似的に同一の評価結果が得られると推測できることから，モデルパラメータの推定が容易な簡易習熟性 T-D SRGM の使用が好ましい．

DS2 について

次に，DS2 に対する式 (2.7) の簡易習熟性 T-D 関数，および式 (2.8) の一般習熟性 T-D 関数で表されるテスト空間占有率の時間的変化を，それぞれ図 4.7 および図 4.8 に示す．DS2 については，一般習熟性 T-D SRGM の推定値が $v_1 > v_2$ であることか

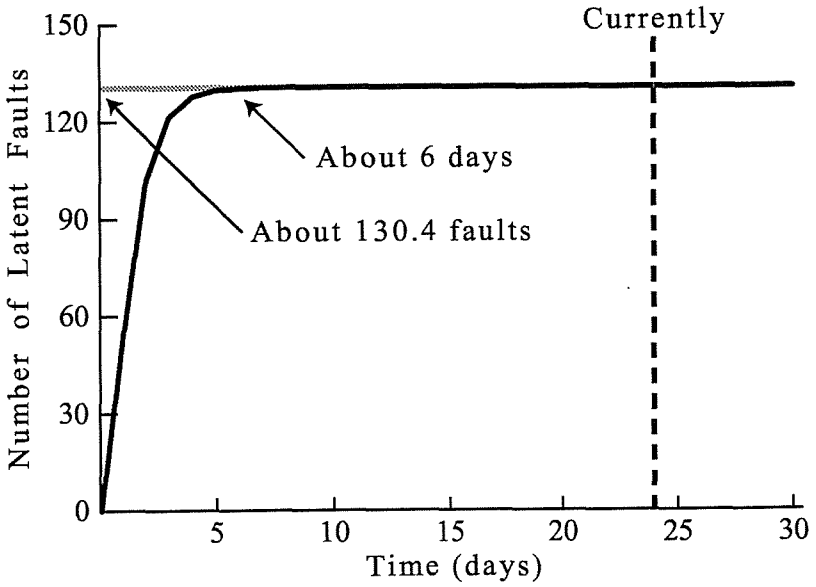

図 4.7: 推定された簡易習熟性 T-D 関数 $u_{bs}(t)$ (DS2).

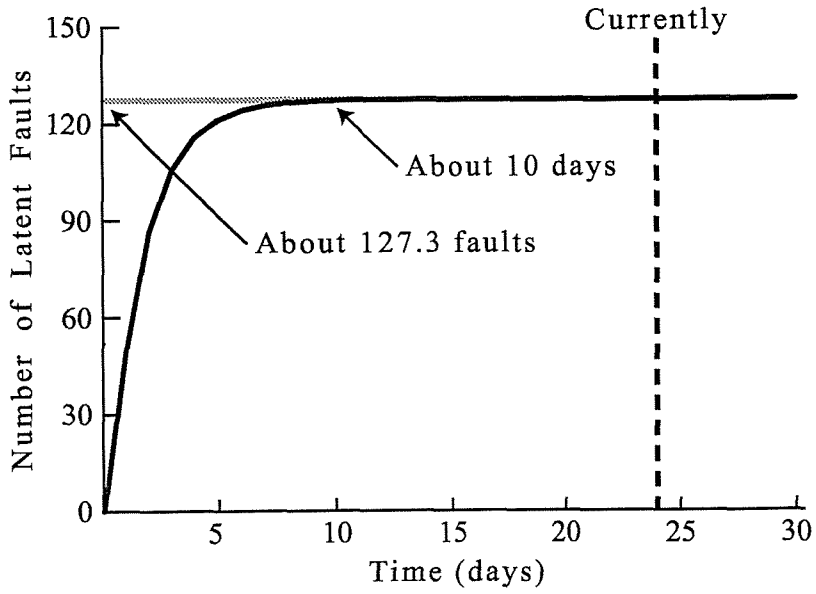

図 4.8: 推定された一般習熟性 T-D 関数 $u_{bG}(t)$ (DS2).

ら，テスト習熟性の低いテストケース設計者によりテストケースが設計されテストしている場合，あるいは一般的なテスト習熟性を有するテストケース設計者が新規開発あるいは部品化されたモジュールの再利用率が低いソフトウェア製品に対するテストケースを設計しテストしている場合が推測できる．つまり，テスト習熟性を表すパラメータ p の推定値が $p \approx 1$ であること，およびテスト空間の初期拡大がないこと，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 の関係が $v_1 \gg v_2$ であることなどを総合的に判断すると，テスト習熟性の低いテストケース設計者によりテストケースが設計され，テストしていると判断できる．また，図4.8からテスト空間占有率が指数関数的に増大していることから，部品化されたモジュールの再利用率の高いソフトウェアシステムであることも推測できる．これは，簡易習熟性T-D SRGMのパラメータ p の推定値が $p = 1$ であること，および図4.7から，同様のことが推測できる．さらに，テスト空間占有率がソフトウェアシステム全体に拡大するまでに要する時間，つまり簡易習熟性T-D 関数では約6日間，一般習熟性T-D 関数では約10日間の差が発生している．これは，一般習熟性T-D 関数において，テスト空間拡大率 v を実際のテスト工程に対する適合性および妥当性を向上させるために，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 は異なる意味を表すモデルパラメータとして厳密に捉えた結果と思われる．

DS3について

また，DS3に対する式(2.7)の簡易習熟性T-D 関数，および式(2.8)の一般習熟性T-D 関数で表されるテスト空間占有率の時間的変化を，それぞれ図4.9および図4.10に示す．DS3については，一般習熟性T-D SRGMの推定値が $v_1 > v_2$ であることか

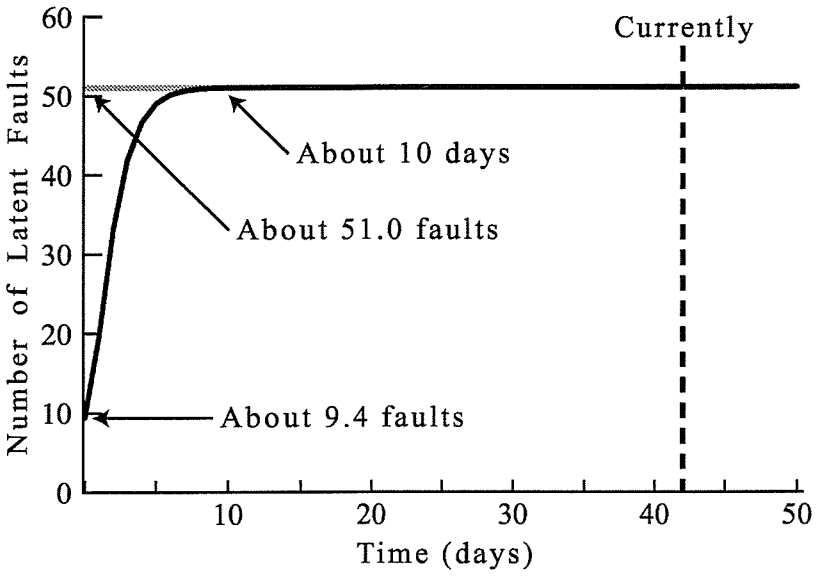

図 4.9: 推定された簡易習熟性 T-D 関数 $u_{b_S}(t)$ (DS3).

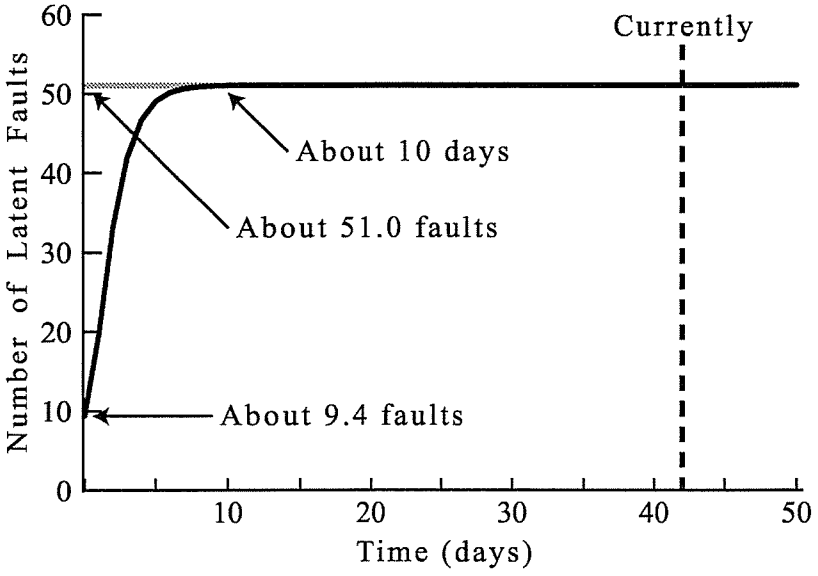

図 4.10: 推定された一般習熟性 T-D 関数 $u_{b_G}(t)$ (DS3).

ら，これは新規開発あるいは部品化されたモジュールの再利用率が低いソフトウェア製品を一般的なテスト習熟性を有するテストケース設計者によりテストケースが設計されテストしている場合であることが推測できる．この場合，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 の関係が $v_1 \gg v_2$ でないこと，およびテスト習熟性を表すパラメータ p の推定値から，一般的なテスト習熟性を有するテストケース設計者がテストケースを設計しテストしている場合と判断できる．これは，図 4.10 において，テスト空間の初期拡大があり，テスト空間占有率が指数関数的に増大していることからも同様のことが伺える．さらに，表 4.2 および表 4.3 の推定結果から，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 の関係が $v_1 \approx v_2$ であること，および両 SRGM のモデルパラメータの推定値がほぼ同一であることが確認できる．また，テスト空間占有率の時間的推移を表す図 4.9 および図 4.10 においても同一の結果が得られている．ゆえに，本データセットの場合も DS1 と同様に，モデルパラメータの少ない簡易習熟性 T-D SRGM を用いた定量的信頼性評価でも，近似的に同一の評価結果が得られることが推測できる．

DS4 について

最後に，DS4 に対する式 (2.7) の簡易習熟性 T-D 関数，および式 (2.8) の一般習熟性 T-D 関数で表されるテスト空間占有率の時間的変化を，それぞれ図 4.11 および図 4.12 に示す．DS4 については，一般習熟性 T-D SRGM の推定値が $v_1 > v_2$ であることから，これは新規開発あるいは部品化されたモジュールの再利用率が低いソフトウェア製品を一般的なテスト習熟性を有するテストケース設計者によりテストケースが設計されテストされている場合であることが推測できる．この場合，パラメー

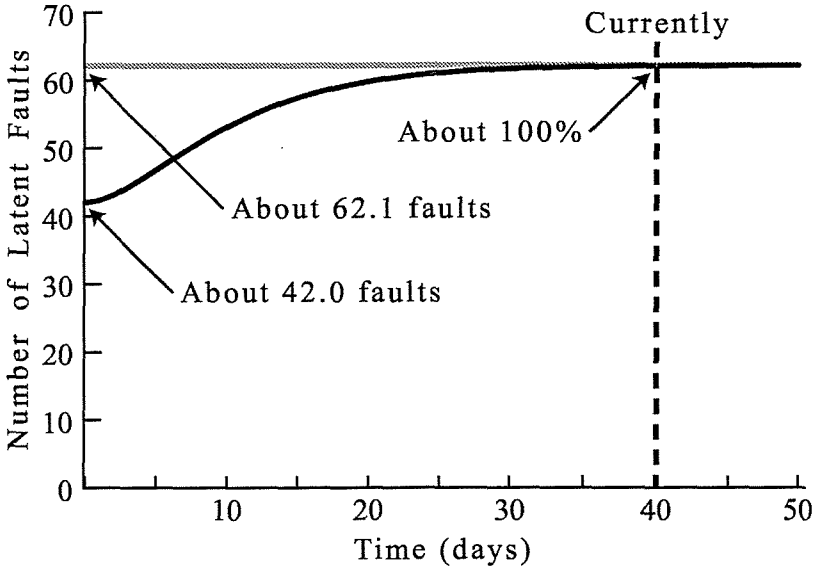

図 4.11: 推定された簡易習熟性 T-D 関数 $u_{b_S}(t)$ (DS4).

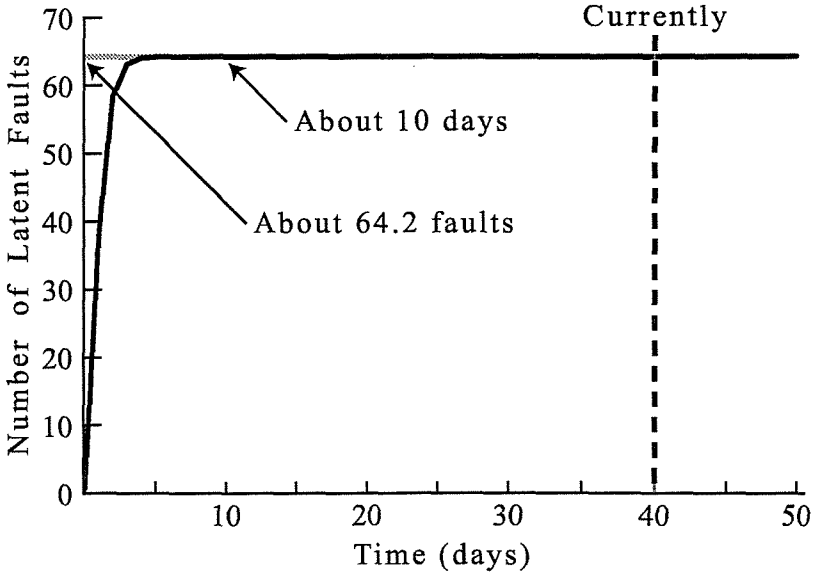

図 4.12: 推定された一般習熟性 T-D 関数 $u_{b_G}(t)$ (DS4).

タ p の推定値として，テスト開始前のテスト習熟性は比較的低い値を示すが，フォールト発見可能領域の拡大率 v_1 およびテスト空間拡大率 v_2 の関係が $v_1 \gg v_2$ でなく，両モデルパラメータの推定値を他のデータセットと比較した場合，両者共に秀でた推定結果であることから，高いテスト習熟性を有するテストケース設計者がテストケースを設計し，テストしていると判断できる．これは，図 4.12 において，テスト空間占有率が指数関数的に増大していることからも同様のことが伺える．さらに，簡易習熟性 T-D SRGM におけるパラメータ p の推定値は，テスト開始前のテスト習熟性が極めて高い $p = 0.32$ を示す．これは，テスト開始直後におけるテストケース設計者の発見可能であるフォールト数が全体の約 70% であると推測できる．また，図 4.11 からテスト空間占有率はＳ字形成長曲線を描き，現在はほぼソフトウェアシステム全体に拡大していることも確認できる．以上のことから，一般習熟性 T-D SRGM と同様の推測ができる．

4.2.2 不完全デバッグ T-D 関数に基づく SRGM

フォールト修正時において新規フォールトの作り込みが把握できる，式 (2.12) の不完全デバッグ T-D 関数で表されるテスト空間占有率の時間的変化を各データセット毎に図 4.13 ～図 4.16 でそれぞれ示す．

DS1 について

DS1 については，表 4.4 の推定結果から $\beta \approx 0$ であることがわかる．これは，テスト開始から現在まで，ソフトウェアシステム内に潜在する総フォールト数は殆ど横這いで一定であることを表し，テストと共に発見されたフォールトを修正および除

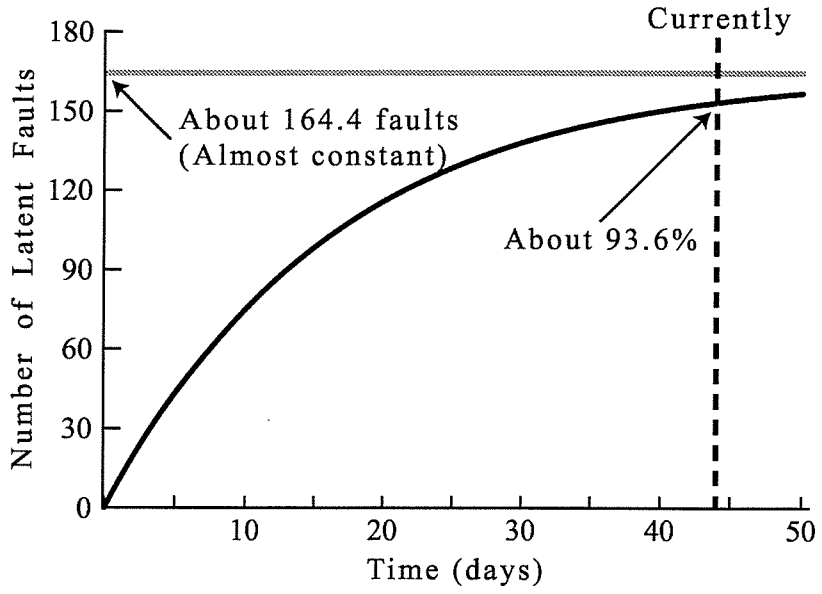

図 4.13: 推定された不完全デバッグ T-D 関数 $u_c(t)$ (DS1).

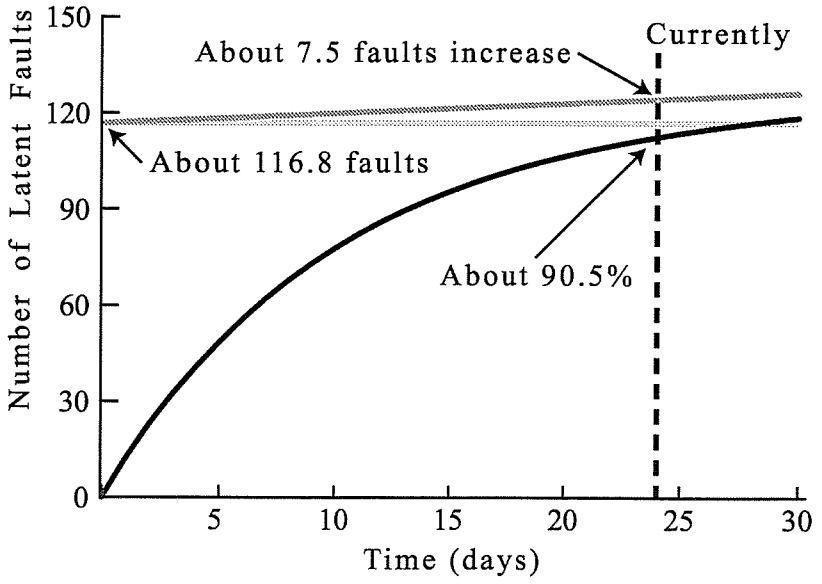

図 4.14: 推定された不完全デバッグ T-D 関数 $u_c(t)$ (DS2).

去する際，新規フォールトの作り込みが無かったことを意味する．つまり本プロジェクトは，部品化されたモジュールの再利用率が高いソフトウェアシステムを一般的なテスト習熟性を有するデバッグ実施者によりテストおよびフォールト修正がなされていることが推測できる．一方，デバッグ作業において，細心の注意を払ったフォールト修正および除去が可能である，高いテスト習熟性を有するデバッグ実施者による作業であることも考えられる．しかし，図 4.13 において，現在に至ってもテスト空間占有率はソフトウェアシステム全体に拡大していないことから，本データセットは一般的なテスト習熟性のデバッグ実施者により採取されたものであることが推測できる．

DS2 について

DS2 については，表 4.4 の推定結果から $\beta > 0$ であること，および図 4.14 からテスト開始から現在までに，デバッグ作業時において新規フォールトが約 7.5 件作り込まれていることがわかる．つまり，テスト開始前に潜在する総期待フォールト数に対するデバッグ作業時における新規フォールト挿入率は約 6.4% であり，約 12.3 件に 1 件の割合で新規フォールトが作り込まれていることを意味する．これは，新規フォールトの作り込まれる割合が比較的小さいことから，テスト対象ソフトウェアシステムは，部品化されたモジュールの再利用率の高いことが伺える．また，新規フォールトの作り込みに対して，テスト空間が追随してソフトウェアシステム全体への拡大に時間を必要としていること，および現在に至ってもテスト空間占有率はソフトウェアシステム全体に拡大していないことなどを総合的に判断すると，テスト習熟性の低いデバッグ実施者により，テストおよびフォールト修正がなされてい

る場合であることが推測できる．

DS3について

DS3については，表4.4の推定結果および図4.15から，テスト終了時点までに，デバッグ作業において新規フォールトが約9.6件作り込まれたことが推測できる．つまり，テスト開始前に潜在する総期待フォールト数に対するデバッグ作業時における新規フォールト挿入率は約22.4%であり，約4.5件に1件の割合で新規フォールトが作り込まれたことを意味し，新規フォールトの作り込まれる割合が比較的大きいことから，テスト対象ソフトウェアシステムは，部品化されたモジュールの再利用率の低いことが伺える．また，図4.15から，テスト開始から約12日間でソフトウェアシステム内におけるテスト空間占有率が約100%に達していることがわかる．

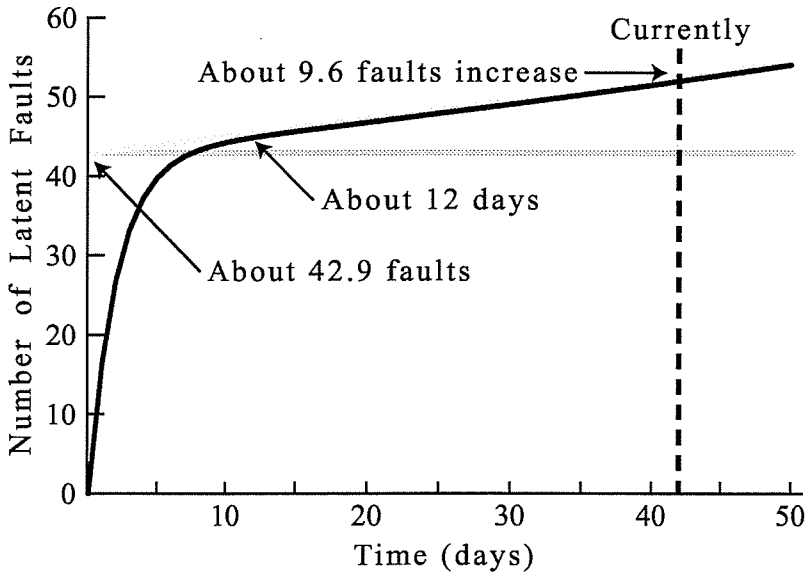

図 4.15: 推定された不完全デバッグT-D関数 $u_c(t)$ (DS3).

その後，テストと共に発見されたフォールトを修正する際，新規フォールトが作り込まれることにより，ソフトウェアシステムおよびテスト空間がそれぞれ拡大している様子，およびフォールト修正時において新規フォールトが作り込まれると共に，それに追随してテスト空間が瞬時に拡大していることが伺える．ゆえに，デバッグ実施者のテスト習熟性は一般的なレベルと推測できる．

DS4について

最後にDS4については，表4.4の推定結果から$\beta \approx 0$であり，テスト開始から現在まで，ソフトウェアシステム内に潜在する総フォールト数は殆ど横這いで一定であることを表し，デバッグ作業において新規フォールトの作り込みが無かったことが推測できる．また図4.16から，テスト空間占有率は指数関数的に増大し，約8日間

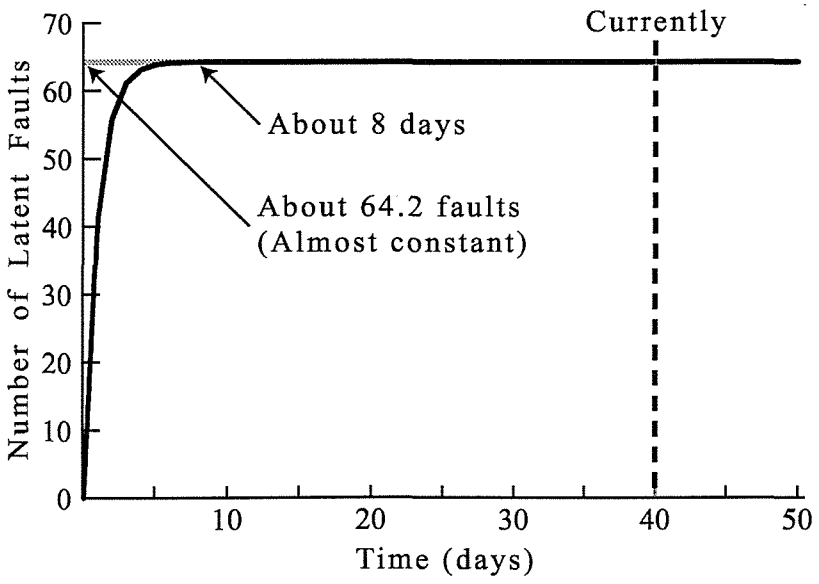

図 4.16: 推定された不完全デバッグT-D関数 $u_c(t)$ (DS4).

でソフトウェアシステム全体に拡大していることが確認できる．ゆえに本プロジェクトは，デバッグ作業において細心の注意を払ったフォールト修正および除去が可能である，テスト習熟性の高いデバッグ実施者によりテストおよびフォールト修正されている場合であることが推測できる．

4.3　信頼性評価尺度の推定と考察

本節では，第2章および第3章で議論したNHPPに基づく4種類のテスト空間依存型SRGMにより導出できる定量的な信頼性評価尺度の推定結果について考察を行う．ここでは，発見可能となる瞬間フォールト増加率，瞬間フォールト発見率，ソフトウェア信頼度および瞬間MTBFの信頼性評価尺度を用いて，これらの推定結果から読み取れる情報に基づいて，今後のテスト管理施策として講じ得る最良の方法について，各データセット毎に議論する．

以下の図中で使用している記号は，基本形T-D SRGM，簡易習熟性T-D SRGM，一般習熟性T-D SRGM，および不完全デバッグT-D SRGMに基づくテスト空間成長関数を$\gamma_a(t)$，$\gamma_{b_S}(t)$，$\gamma_{b_G}(t)$および$\gamma_c(t)$，強度関数を$h_a(t)$，$h_{b_S}(t)$，$h_{b_G}(t)$および$h_c(t)$，ソフトウェア信頼度を$R_a(x|t)$，$R_{b_S}(x|t)$，$R_{b_G}(x|t)$および$R_c(x|t)$，および瞬間MTBFを$MTBF_{\mathrm{I}\,a}(t)$，$MTBF_{\mathrm{I}\,b_S}(t)$，$MTBF_{\mathrm{I}\,b_G}(t)$および$MTBF_{\mathrm{I}\,c}(t)$でそれぞれ表現する．

DS1について

先ず最初に，式(2.15)–式(2.18)のテスト空間成長関数で表される，テストにより発見可能となる単位テスト時間当りのフォールトの増加数，つまり発見可能となる瞬

間フォールト増加率の時間的変化を図4.17に示す．図4.17から，$\gamma_a(t)$および$\gamma_c(t)$の場合，テストにより発見可能となるフォールト数はテスト開始直後に最大値を示し，テストの進捗と共に減少していることから，順調にソフトウェア信頼度が成長している様子が伺える．また，$\gamma_{b_S}(t)$および$\gamma_{b_G}(t)$の場合，テストにより発見可能となるフォールト数はテスト開始直後から徐々に増加し，約15日辺りで最大値を示し，その後減少に転じていることがわかる．これは，簡易習熟性T-D SRGMおよび一般習熟性T-D SRGMを構築する際，テストケース設計者のテスト習熟性がフォールト発見事象に影響するものと仮定していることから，テストケース設計者がテスト対象ソフトウェアシステムの要求仕様書および内部構造に対する理解度合を表している．つまり，テスト開始から約15日当り迄，順調にテスト対象物に対する理解度が向上し，15日目以降でも向上速度は低下しているが，現在も向上し続けている

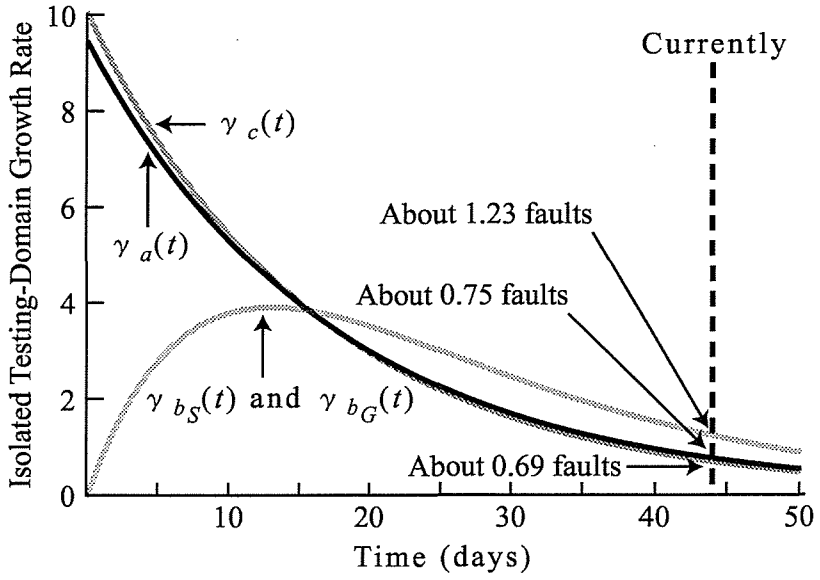

図 4.17: 推定されたテスト空間成長関数 (DS1).

様子が伺える．さらに，現時点 $(t = 44)$ で発見可能となる瞬間フォールト増加率は，それぞれのSRGMにおいて，

$\gamma_a(44) = 0.75$ (件/日)，　$\gamma_{b_S}(44) = \gamma_{b_G}(44) = 1.23$ (件/日)，　$\gamma_c(44) = 0.69$ (件/日)，

となり，テストケース設計者のテスト習熟性を考慮した簡易習熟性T-D SRGMおよび一般習熟性T-D SRGMの推定値が厳しい値であることが確認できる．

次に，式(3.11)–式(3.14)の強度関数で表される単位テスト時間当りに発見されるフォールト数，つまり瞬間フォールト発見率の時間的変化を図4.18に示す．図4.18から，$h_a(t)$ および $h_c(t)$ の場合，テストの進捗と共に単位テスト時間当りに発見されるフォールト数は徐々に増加し，テスト開始から約15日目辺りで最大値を示し，以降，減少に転じていることから，多くのフォールトが発見され順調にソフトウェア信頼度が成長している様子が伺える．また，$h_{b_S}(t)$ および $h_{b_G}(t)$ の場合，テスト開始直後の単位テスト時間当りのフォールト発見率は約1.86(件/日)を示し，2日目は約1.60(件/日)と減少していることがわかる．これは，この時期に何らかの手戻り作業，つまり仕様変更あるいは仕様追加などが発生したことが考えられる．その後，単位テスト時間当りに発見されるフォールト数は増加に転じ，約20日目辺りから再び減少に転じていることから，図4.17で示した信頼性評価結果と同様の状態であることが確認できる．

さらに，同一テスト環境下において任意時間区間でソフトウェア故障が発生しない度合，つまり式(3.19)のソフトウェア信頼度をそれぞれのSRGMにより推定した結果を図4.19に示す．図4.19から，現時点 $(t = 44)$ においてテストを継続実施し，1日間ソフトウェア故障が発生しない確率は，

$$R_a(1 \mid 44) = R_c(1 \mid 44) = 15.4\ \%, \qquad R_{b_S}(1 \mid 44) = R_{b_G}(1\ 1 \mid 44) = 16.8\ \%,$$

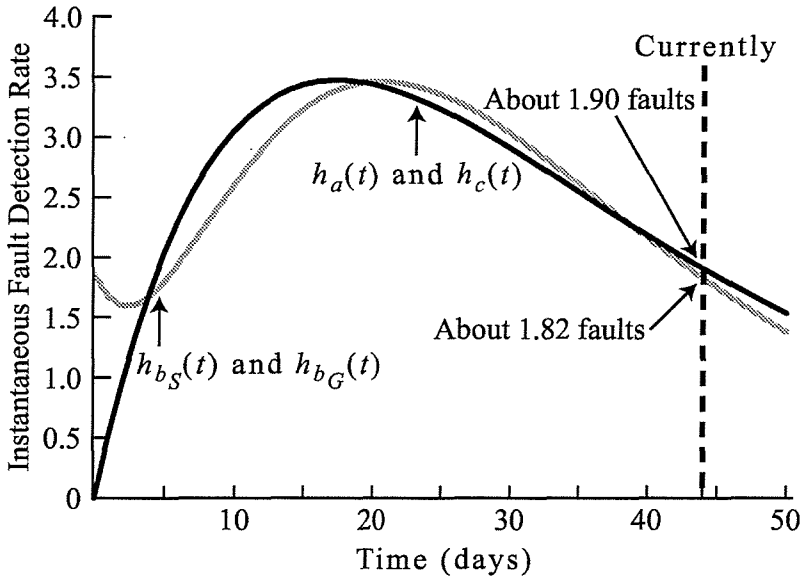

図 4.18: 推定された強度関数 (DS1).

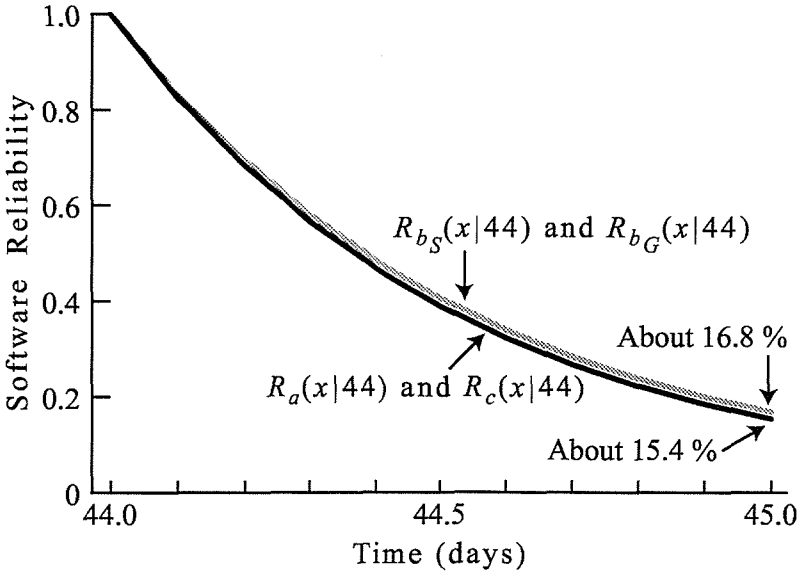

図 4.19: 推定されたソフトウェア信頼度 (DS1).

であることが推測できる．これは，約83～85 %の確率でソフトウェア故障が発生することを意味する．このソフトウェア信頼度を用いて，実際のユーザ運用環境に適応させるためにフォールト発見率の厳しさ係数[21]を導入して，テスト環境とユーザ運用環境のソフトウェア実行における相関関係を考慮したうえで，実際のユーザ運用を想定した環境下で1年間運用した場合のソフトウェア信頼度を算出した結果，

$$R_a(365 \mid 44) = R_c(365 \mid 44) = 0.19\ \%, \quad R_{b_S}(365 \mid 44) = R_{b_G}(365 \mid 44) = 1.14\ \%,$$

となる．これは，実際のユーザ運用環境下で1年間使用した場合，約99 %の確率でソフトウェア故障が発生することを表しており，現時点では，ユーザ運用に耐え得るソフトウェア信頼度でないことが確認できる．

また，式(3.21)の瞬間MTBF，つまり任意のテスト時刻における平均ソフトウェア故障時間間隔をそれぞれのSRGMにより推定した結果を図4.20に示す．図4.20

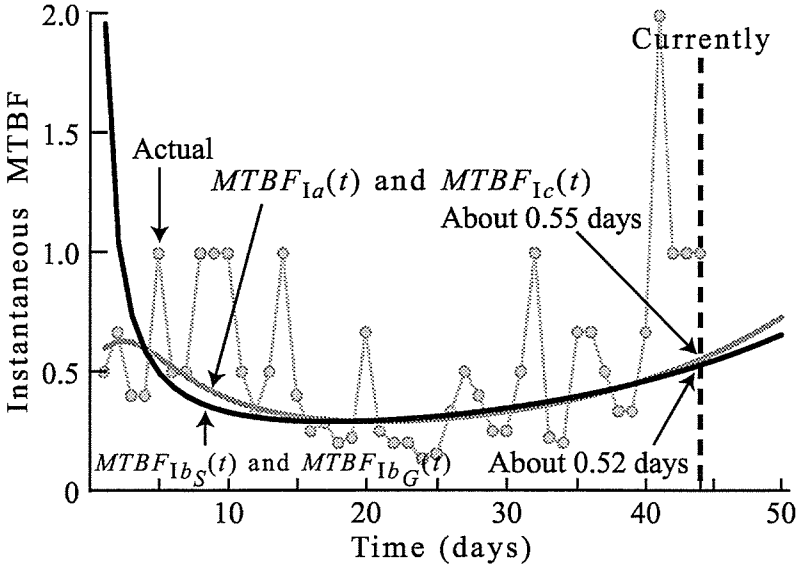

図 4.20: 推定された瞬間MTBF (DS1).

において，全てのSRGM共に瞬間MTBF値は約0.5 (日) を表し，ソフトウェア故障が発生すること無く動作可能な時間が約0.5 (日) であると推測できる．これは，図4.19で示した信頼性評価結果と同様のことが伺え，ソフトウェア信頼度の推定値を裏付ける結果になっている．さらに，本評価尺度においても，テストケース設計者のテスト習熟性を考慮した簡易習熟性T-D SRGMおよび一般習熟性T-D SRGMの推定値が厳しい値であることが確認できる．

最後に，DS1に対するテスト管理施策として，テスト空間を考慮した4種類のSRGMによる定量的信頼性評価の結果，

- 瞬間フォールト増加率 < 瞬間フォールト発見率
  瞬間フォールト発見率 > 1.0 (件/日) で低レベル
- 1日後のソフトウェア信頼度は約15 % で低レベル
- 瞬間MTBFは約0.55 (日) で低レベル

であることから，継続したテストが必要であると考えられる．さらに，テスト空間占有率を表す図4.5，図4.6および図4.13から，テスト空間が100 % に達していないことから，テスト未実施であるモジュールおよび機能の存在が推測でき，異なる観点でテストケースを新規に設計する必要があることを示唆している．また，デバッグ作業中に新規フォールトの作り込みが殆ど無いと予測されていることから，DS1の場合，テストケース設計者のテスト習熟性を考慮した簡易習熟性T-D SRGMおよび一般習熟性T-D SRGMを用いた信頼性評価が妥当であると思われる．これは，図4.17および図4.18による信頼性評価結果と同一であり，お互いの信頼性評価を裏付ける結果であることが確認できる．

DS2について

式(2.15)–式(2.18)のテスト空間成長関数で表される，テストにより発見可能となる単位テスト時間当りのフォールトの増加数，つまり発見可能となる瞬間フォールト増加率の時間的変化を図4.21に示す．図4.21から，$\gamma_a(t)$および$\gamma_c(t)$の場合，テストにより発見可能となるフォールト数はテスト開始直後にそれぞれ最大値を示し，テストの進捗と共に減少していることがわかる．また，$\gamma_{b_S}(t)$および$\gamma_{b_G}(t)$の場合，テストにより発見可能となるフォールト数は急速に増加し，テスト開始から約1日で最大値を示し，その後減少に転じていることがわかる．これは，テストケース設計者のテスト習熟性が急速に向上している様子を表しており，図4.7および図4.8において，テスト空間占有率が指数関数的に増大し，短期間でソフトウェアシステム全体にまで拡大していることからも同様のことが伺える．さらに，現時点($t = 24$)

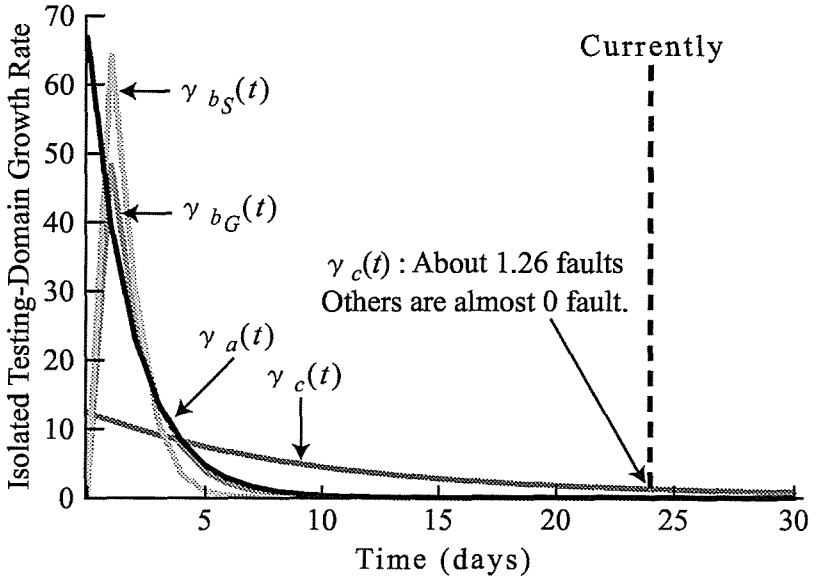

図 4.21: 推定されたテスト空間成長関数 (DS2).

で発見可能となる瞬間フォールト増加率は，それぞれのSRGMにおいて，

$$\gamma_a(24) = \gamma_{b_S}(24) = \gamma_{b_G}(24) = 0 \text{ (件/日)}, \qquad \gamma_c(24) = 1.26 \text{ (件/日)},$$

となり，デバッグ作業中において新規フォールトの作り込まれる可能性を考慮した不完全デバッグT-D SRGMの推定値が厳しい値であることが確認できる．

次に，式(3.11)–式(3.14)の強度関数で表される単位テスト時間当りに発見されるフォールト数，つまり瞬間フォールト発見率の時間的変化を図4.22に示す．図4.22から，$h_a(t)$，$h_{b_S}(t)$，$h_{b_G}(t)$および$h_c(t)$共に類似した挙動を示すことが確認できる．つまり，テスト開始から約5日間，単位テスト時間当りに発見されるフォールト数は増加し，その後減少に転じていることから，順調にフォールトが発見されており，ソフトウェア信頼度が成長している様子が伺える．また，現時点($t = 24$)における瞬間フォールト発見率はそれぞれ

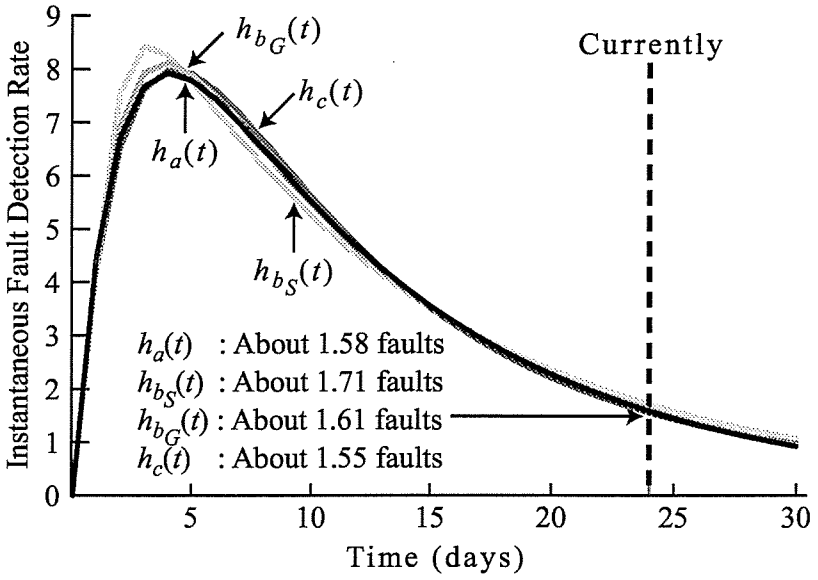

図 4.22: 推定された強度関数 (DS2).

$$h_a(24)=1.58\ (\text{件/日}), \qquad h_{b_S}(24)=1.71\ (\text{件/日}),$$
$$h_{b_G}(24)=1.61\ (\text{件/日}), \qquad h_c(24)=1.55\ (\text{件/日}),$$

であることがわかる．この結果は，テストケース設計者のテスト習熟性を考慮した簡易習熟性 T-D SRGM の推定値が厳しい値であることが確認できる．

また，式 (3.19) のソフトウェア信頼度をそれぞれの SRGM により推定した結果を図 4.23 に示す．図 4.23 から，現時点 ($t = 24$) において，同一テスト環境下で 1 日間ソフトウェア故障が発生しない確率は，

$$R_a(1\,|\,24) = R_{b_G}(1\,|\,24) = R_c(1\,|\,24) = 22.0\ \%, \qquad R_{b_S}(1\,|\,24) = 19.3\ \%,$$

であることが推測できる．これは，信頼性評価に用いた SRGM に関係なく約 80 % 近い確率でソフトウェア故障が発生することを表している．このソフトウェア信頼度を用いて，実際のユーザ運用環境に適応させるためにフォールト発見率の厳しさ

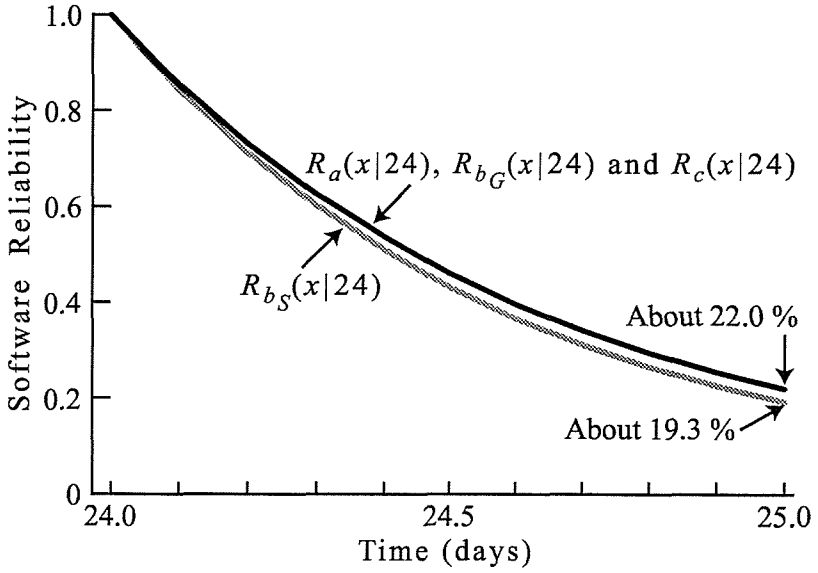

図 4.23: 推定されたソフトウェア信頼度 (DS2).

係数 [21] を導入して，テスト環境とユーザ運用環境のソフトウェア実行における相関関係を考慮したうえで，実際のユーザ運用を想定した環境下で 1 年間運用した場合のソフトウェア信頼度を算出した結果，

$$R_a(365 \mid 24) = 30.5\ \%, \qquad R_{b_S}(365 \mid 24) = 21.2\ \%,$$
$$R_{b_G}(365 \mid 24) = 28.7\ \%, \qquad R_c(365 \mid 24) = 36.2\ \%,$$

となる．これは，実際のユーザ運用環境下で 1 年間使用した場合，約 65 ～ 80 % の確率でソフトウェア故障が発生することを表しており，ユーザ環境下で運用した場合，発生したソフトウェア故障の修正・除去に費やす保守コストが多大になることが考えられる．この結果は，図 4.22 で示した信頼性評価結果と同様，テストケース設計者のテスト習熟性を考慮した簡易習熟性 T-D SRGM の推定値が厳しい値であ

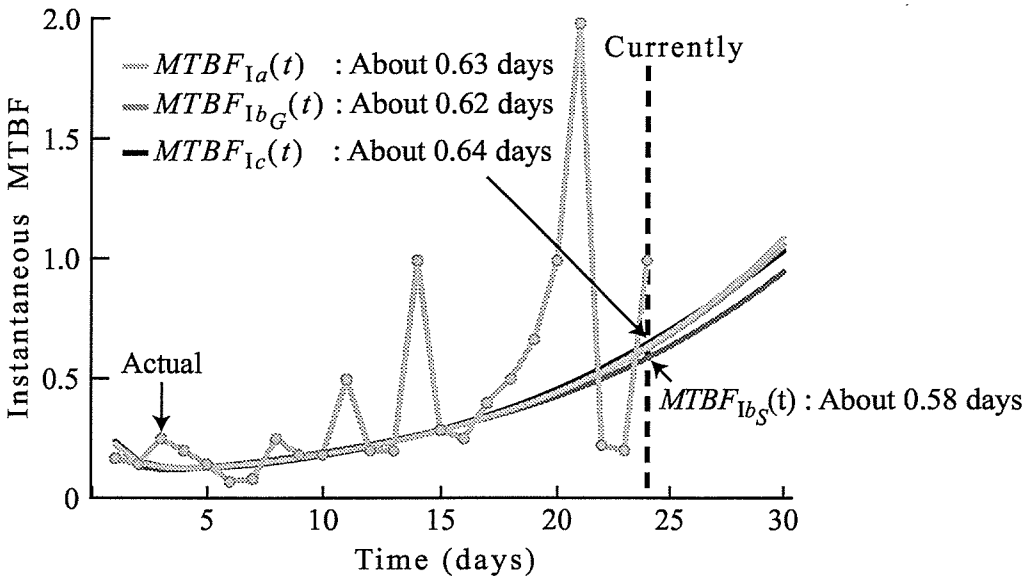

図 4.24: 推定された瞬間 MTBF (DS2).

ることが確認できる．

さらに，式(3.21)の瞬間MTBFをそれぞれのSRGMにより推定した結果を図4.24に示す．図4.24から，信頼性評価に用いたSRGMに関係なく約0.6 (日) であることがわかる．また，現時点 ($t = 24$) において同一環境下でテストを継続することにより，瞬間MTBF値が向上する様子も伺える．この結果は，前述で示した図4.22および図4.23の信頼性評価結果と同様，テストケース設計者のテスト習熟性を考慮した簡易習熟性T-D SRGMの推定値が厳しい値であることが確認できる．

最後に，DS2に対するテスト管理施策として，定量的な信頼性評価結果より，

- 瞬間フォールト増加率 < 瞬間フォールト発見率
  瞬間フォールト発見率 > 1.0 (件/日) で低レベル
- 1日後のソフトウェア信頼度は約20 %前後で低レベル
  ユーザ環境下での運用を考慮した場合でも約30 %前後で低レベル
- 瞬間MTBFは約0.6 (日) で低レベル

であることから，継続したテストが必要であると考えられる．さらに，完全デバッグ環境を考慮したテスト空間占有率を表す図4.7および図4.8において，テスト工程初期においてテスト空間占有率は100 %に達しているのに対して，不完全デバッグ環境を考慮したテスト空間占有率を表す図4.14では，いまだにテスト空間占有率が約90.5 %であることから，デバッグ作業中のフォールト修正において，新規フォールトがテスト済のモジュールおよび機能に関連する箇所に作り込まれたことが考えられる．ゆえに，フォールト修正に関連するモジュールおよび機能の再テストが必要であることを示唆している．また，デバッグ作業中に新規フォールトの作り込み

が予測されていることから，DS2の場合，フォールト修正時における新規フォールト混入可能性を考慮した不完全デバッグT-D SRGMを用いた信頼性評価が妥当であると思われる．また，図4.22，図4.23および図4.24による定量的な信頼性評価結果から，厳しい推定結果を示すテストケース設計者のテスト習熟性を考慮した簡易習熟性T-D SRGMを用いた定量的信頼性評価も妥当であると思われる．

DS3について

式(2.15)–式(2.18)のテスト空間成長関数で表される，単位テスト時間当りに発見可能となるフォールトの増加数，つまり発見可能となる瞬間フォールト増加率の時間的変化を図4.25に示す．図4.25から，$\gamma_a(t)$および$\gamma_c(t)$の場合，テストにより発見可能となるフォールト数はテスト開始直後にそれぞれ最大値を示し，テストの進

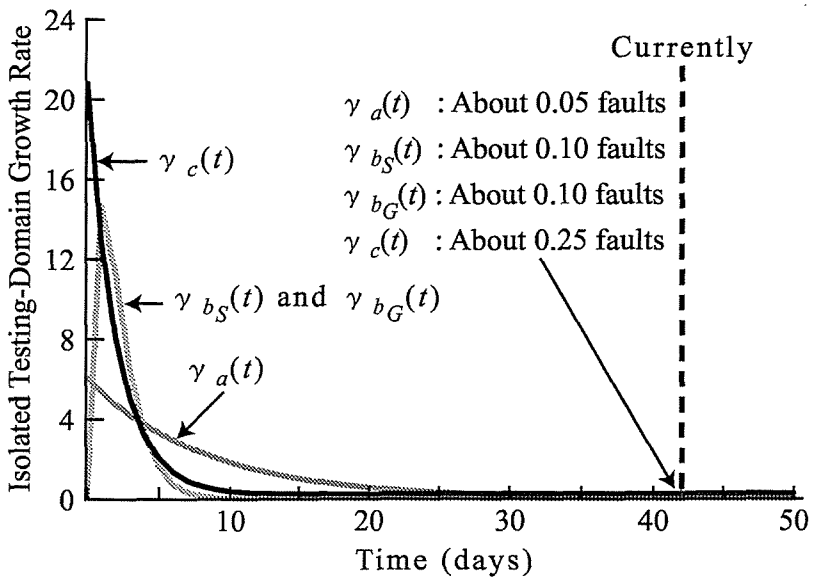

図 4.25: 推定されたテスト空間成長関数 (DS3).

捗と共に減少していることがわかる．また，$\gamma_{b_S}(t)$ および $\gamma_{b_G}(t)$ の場合，テストにより発見可能となるフォールト数は急速に増加し，テスト開始から約1日で最大値を示し，その後減少に転じている．これは，テストケース設計者のテスト習熟性が急速に向上している様子を表しており，図4.9および図4.10において，テスト開始直後にテスト空間の初期拡大があり，テスト空間占有率も指数関数的に増大し，短期間でソフトウェアシステム全体にまで拡大していることからも同様のことが伺える．さらに，現時点 $(t = 42)$ において発見可能となる瞬間フォールト増加率は，それぞれのSRGMにおいて，

$$\gamma_a(42) = 0.05 \text{ (件/日)}, \ \gamma_{b_S}(42) = \gamma_{b_G}(42) = 0.10 \text{ (件/日)}, \ \gamma_c(42) = 0.25 \text{ (件/日)},$$

となり，デバッグ作業中において新規フォールトの作り込まれる可能性を考慮した不完全デバッグT-D SRGMの推定値が厳しい値であることが確認できる．

次に，式(3.11)–式(3.14)の強度関数で表される単位テスト時間当りに発見されるフォールト数，つまり瞬間フォールト発見率を図4.26に示す．図4.26から，テスト工程中期辺り迄は，$h_a(t)$，$h_{b_S}(t)$，$h_{b_G}(t)$ および $h_c(t)$ 共に類似した挙動を示すことが確認できる．つまり，テスト開始から約5日間，単位テスト時間当りに発見されるフォールト数は増加し，その後減少に転じていることから，順調にフォールトが発見されており，ソフトウェア信頼度が成長している様子が伺える．また，現時点 $(t = 42)$ における瞬間フォールト発見率はそれぞれのSRGMにおいて，

$$h_a(42) = 0.06 \text{ (件/日)}, \ h_{b_S}(42) = h_{b_G}(42) = 0.10 \text{ (件/日)}, \ h_c(42) = 0.27 \text{ (件/日)},$$

であることがわかる．これは，図4.25で示した信頼性評価結果と同様，フォールト修正時における新規フォールト混入可能性を考慮した不完全デバッグT-D SRGMの

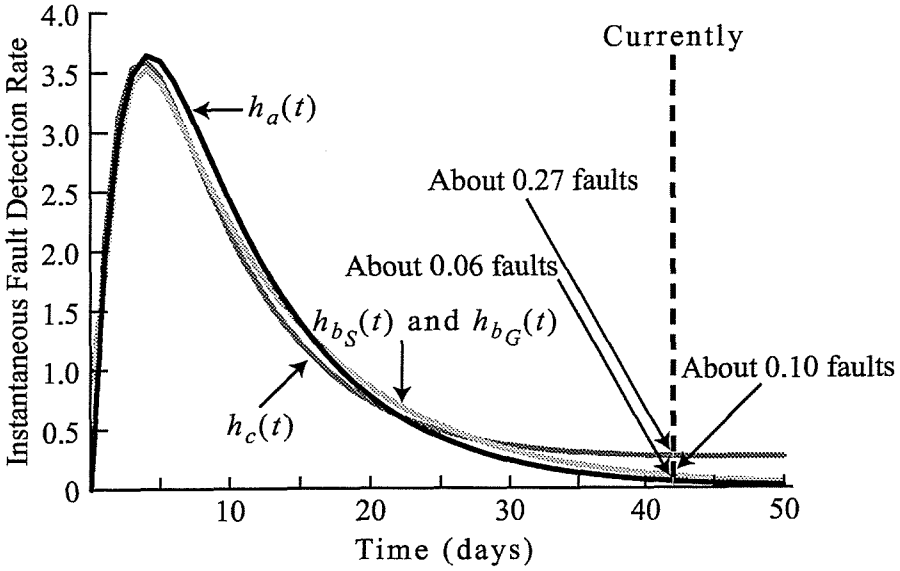

図 4.26: 推定された強度関数 (DS3).

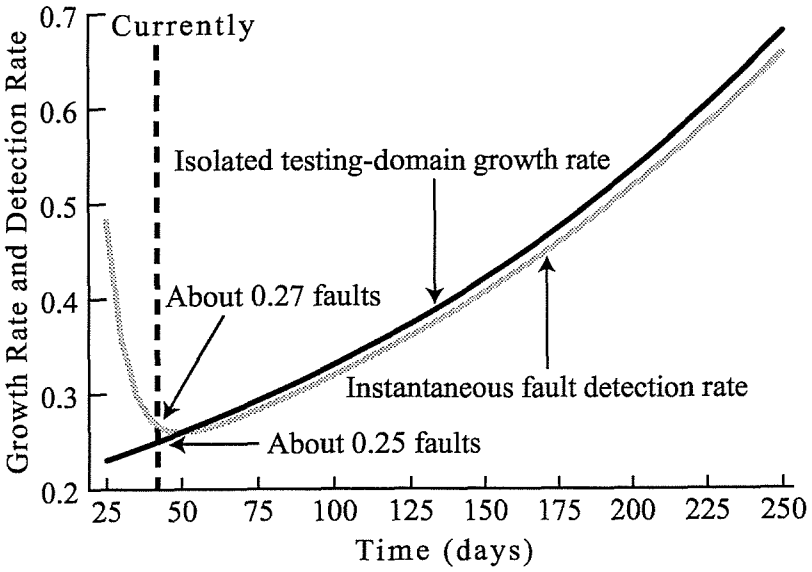

図 4.27: 推定されたテスト空間成長関数および強度関数 (DS3).

推定値が厳しい値であることが確認できる．ここで$h_c(t)$の推定値から，他のSRGMを用いた推定値より瞬間フォールト発見率の減少率が小さいことに注目して，長期間テストを実施した場合の発見可能となる瞬間フォールト増加率を表すテスト空間成長関数，および瞬間フォールト発見率を表す強度関数の推定結果を図4.27に示す．図4.27からも，今後発見されるフォールトは，多数のモジュールと複雑に関連することが予測でき，フォールト修正時に新規フォールトの作り込まれる割合が増加している様子が伺える．

また，式(3.19)のソフトウェア信頼度をそれぞれのSRGMにより推定した結果を図4.28に示す．図4.28から，現時点$(t = 42)$において，同一テスト環境下で1日間ソフトウェア故障が発生しない確率は，

$$R_a(1 \mid 42) = 94.9\ \%, \quad R_{b_S}(1 \mid 42) = R_{b_G}(1 \mid 42) = 90.9\ \%, \quad R_c(1 \mid 42) = 76.6\ \%,$$

であることが推測できる．このソフトウェア信頼度を用いて，実際のユーザ運用環境に適応させるためにフォールト発見率の厳しさ係数[21]を導入して，テスト環境とユーザ運用環境のソフトウェア実行における相関関係を考慮したうえで，実際のユーザ運用を想定した環境下で1年間運用した場合のソフトウェア信頼度を算出した結果，

$$R_a(365 \mid 42) = 99.6\ \%, \qquad R_c(365 \mid 42) = 89.5\ \%,$$
$$R_{b_S}(365 \mid 42) = R_{b_G}(365 \mid 42) = 98.3\ \%,$$

となる．これは，実際のユーザ運用環境下で1年間使用した場合，ソフトウェア故障が発生することなく運用可能な確率が約90 %以上であることを表しており，実際のユーザ運用に耐え得るソフトウェア信頼度であると考えられる．本評価尺度に

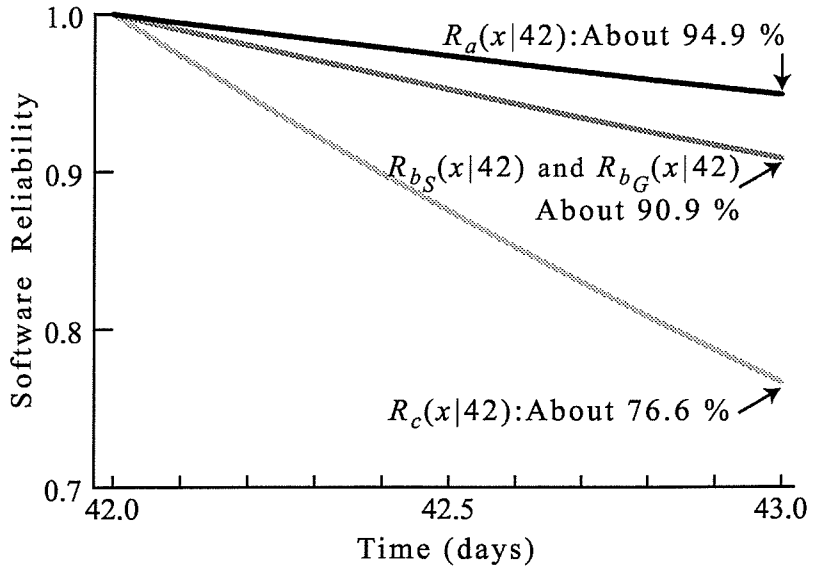

図 4.28: 推定されたソフトウェア信頼度 (DS3).

よる結果も，前述の信頼性評価結果と同様に，新規フォールト混入可能性を考慮した不完全デバッグ T-D SRGM の推定値が厳しい値であることが確認できる．

さらに，式 (3.21) で表される瞬間 MTBF について，それぞれの SRGM により推定した結果を図 4.29 に示す．図 4.29 から，現時点 $(t = 42)$ における瞬間 MTBF の推定値はそれぞれ，

$$MTBF_{\mathrm{I}\,a}(42) = 17.9\ (\text{日}), \qquad MTBF_{\mathrm{I}\,c}(42) = 3.37\ (\text{日}),$$
$$MTBF_{\mathrm{I}\,b_S}(42) = MTBF_{\mathrm{I}\,b_G}(42) = 9.93\ (\text{日}),$$

であることがわかる．また，他の SRGM の推定値とは異なり，新規フォールト混入可能性を考慮した不完全デバッグ T-D SRGM の推定値では，テスト開始から約 40 日目以降において瞬間 MTBF の推定値がほぼ横這いになる．これは，図 4.26 と同様，今後発見されるフォールトの修正は複雑になり，新規フォールトの作り込まれ

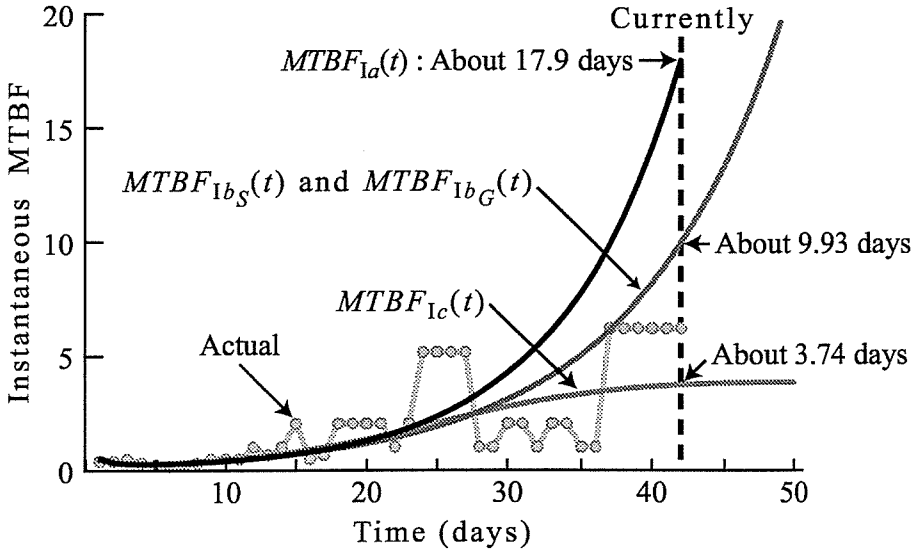

図 4.29: 推定された瞬間 MTBF (DS3).

る可能性が増大することを示唆しており，瞬間 MTBF の推定値が横這いになったものと考えられる.

最後に，DS3 に対するテスト管理施策として，全ての信頼性評価尺度で厳しい値を示すフォールト修正時における新規フォールト混入可能性を考慮した不完全デバッグ T-D SRGM を用いた定量的信頼性評価結果より，

- 瞬間フォールト増加率 ≈ 瞬間フォールト発見率

  瞬間フォールト発見率 < 0.3 (件/日) で高レベル

- 1 日後のソフトウェア信頼度は約 75 % 以上

  ユーザ環境下での運用を考慮した場合でも約 90 % で高レベル

- 瞬間 MTBF は約 3.74 (日) で横這い

であり，残存フォールト数も約 3 件以下であることから，出荷可能なレベルである

と判断できる．今後，残存フォールトを発見するためにテストを継続するか否かについては，開発コストおよび製品出荷スケジュールを考慮した判断が必要である．

DS4について

式(2.15)–式(2.18)のテスト空間成長関数で表される，単位テスト時間当りに発見可能となるフォールトの増加数，つまり発見可能となる瞬間フォールト増加率について$\gamma_a(t)$，$\gamma_{b_G}(t)$および$\gamma_c(t)$の推定結果を図4.30，および$\gamma_{b_S}(t)$の推定結果を図4.31にそれぞれ示す．図4.30および図4.31から，他のデータセットと同様に，$\gamma_a(t)$および$\gamma_c(t)$の場合，テストにより発見可能となるフォールト数はテスト開始直後にそれぞれ最大値を示し，テストの進捗と共に減少していることがわかる．また$\gamma_{b_S}(t)$の場合，テストにより発見可能となるフォールト数は徐々に増加し，テスト開始か

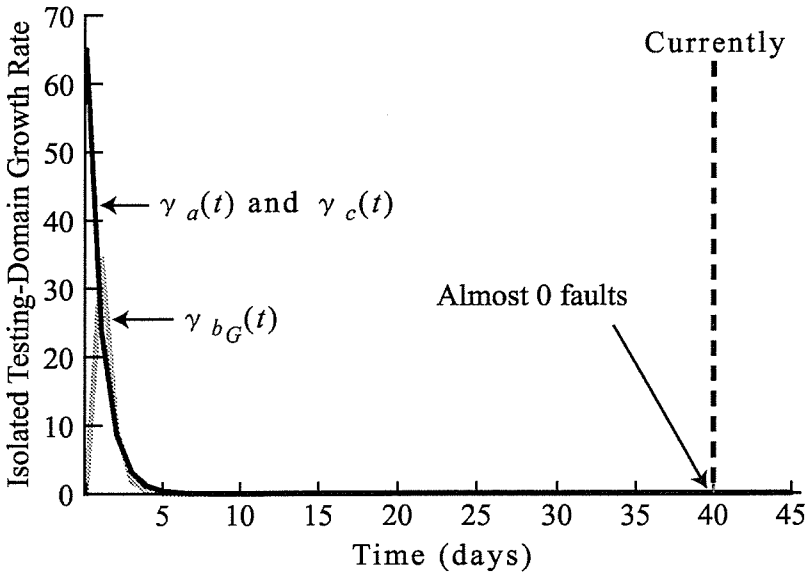

図 4.30: 推定されたテスト空間成長関数 (DS4) (その１).

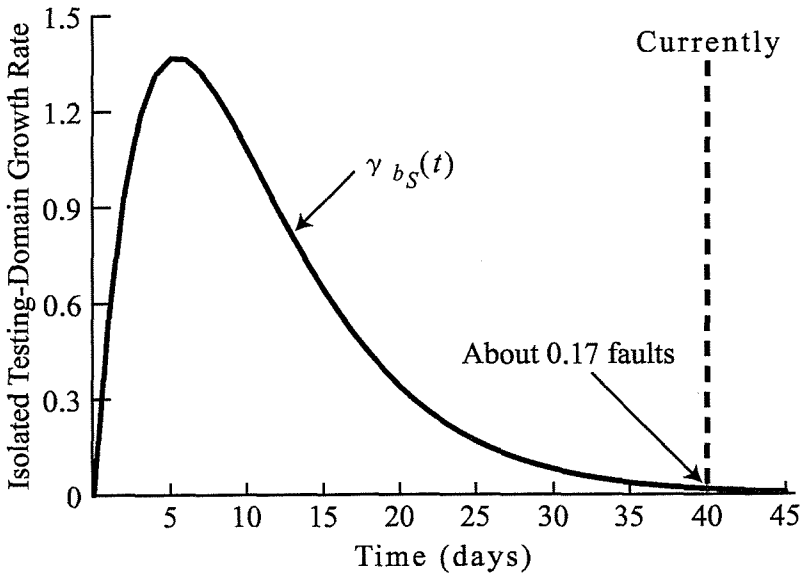

図 4.31: 推定されたテスト空間成長関数 (DS4) (その 2).

ら約 5 日で最大値を示し，その後減少に転じている．同様に $\gamma_{b_G}(t)$ の場合，テストにより発見可能となるフォールト数は急速に増加し，約 1 日で最大値を示し，その後減少に転じている．これは，テストケース設計者のテスト習熟性が向上していることを意味し，テスト習熟性の向上速度の違いが図 4.11 および図 4.12 に示すテスト空間占有率の挙動の違いに現れている．さらに，現時点 $(t = 40)$ で発見可能となる瞬間フォールト増加率は，それぞれの SRGM において，

$$\gamma_a(40) = \gamma_{b_G}(40) = \gamma_c(40) = 0 \text{ (件/日)}, \qquad \gamma_{b_S}(40) = 0.17 \text{ (件/日)},$$

となり，テストケース設計者のテスト習熟性を考慮した簡易習熟性 T-D SRGM の推定値が厳しい値であることが確認できる．

次に，式 (3.11)–式 (3.14) の強度関数で表される単位テスト時間当りに発見されるフォールト数，つまり瞬間フォールト発見率の時間的変化を図 4.32 に示す．図 4.32

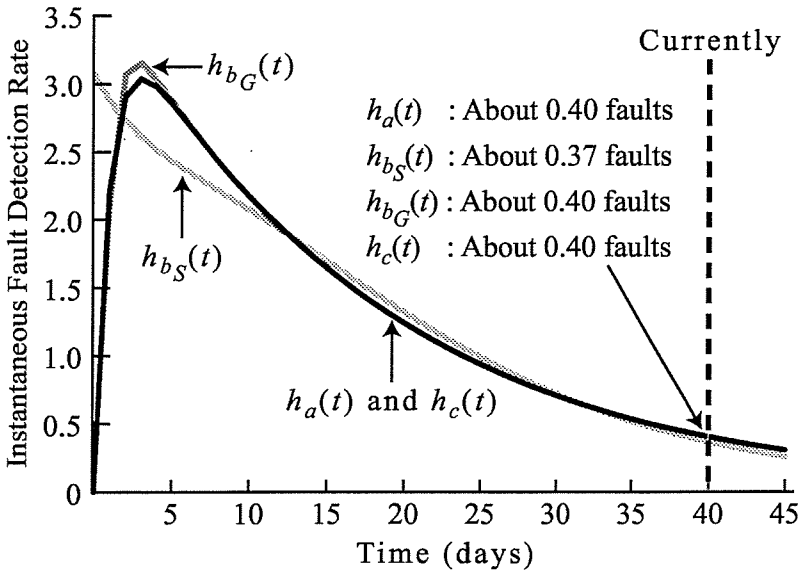

図 4.32: 推定された強度関数 (DS4).

から，$h_a(t)$，$h_{b_G}(t)$ および $h_c(t)$ の挙動は，瞬間フォールト発見率の最大値の違いはあるが，殆ど同じ推定結果を示すことがわかる．ただ，$h_{b_S}(t)$ のみ異なる挙動を表し，テスト開始直後に最大値を示し，順調に瞬間フォールト発見率が減少している様子が伺える．また，現時点 $(t = 40)$ における瞬間フォールト発見率はそれぞれの SRGM において，

$$h_a(40) = h_{b_G}(40) = h_c(40) = 0.40 \text{ (件/日)}, \qquad h_{b_S}(40) = 0.37 \text{ (件/日)},$$

であることがわかる．この結果は，発見可能となる瞬間フォールト増加率による信頼性評価結果とは逆に，テストケース設計者のテスト習熟性を考慮した簡易習熟性 T-D SRGM を除く SRGM の推定値が厳しい値であることが確認できる．

また，式 (3.19) のソフトウェア信頼度をそれぞれの SRGM により推定した結果を図 4.33 に示す．図 4.33 から，現時点 $(t = 40)$ において，同一テスト環境下で 1 日間

ソフトウェア故障が発生しない確率は,

$$R_a(1\,|\,40) = R_{b_G}(1\,|\,40) = R_c(1\,|\,40) = 67.5\ \%, \qquad R_{b_S}(1\,|\,40) = 70.2\ \%,$$

であることが推測できる．このソフトウェア信頼度を用いて，実際のユーザ運用環境に適応させるためにフォールト発見率の厳しさ係数 [21] を導入して，テスト環境とユーザ運用環境のソフトウェア実行における相関関係を考慮したうえで，実際のユーザ運用を想定した環境下で1年間運用した場合のソフトウェア信頼度を算出した結果,

$$R_a(365\,|\,40) = R_{b_G}(365\,|\,40) = R_c(365\,|\,40) = 67.1\ \%, \quad R_{b_S}(365\,|\,40) = 77.3\ \%,$$

となる．これは，実際のユーザ運用環境下で1年間使用した場合，ソフトウェア故障が発生することなく運用可能な確率が約 70 % 前後であることが推測でき，実際

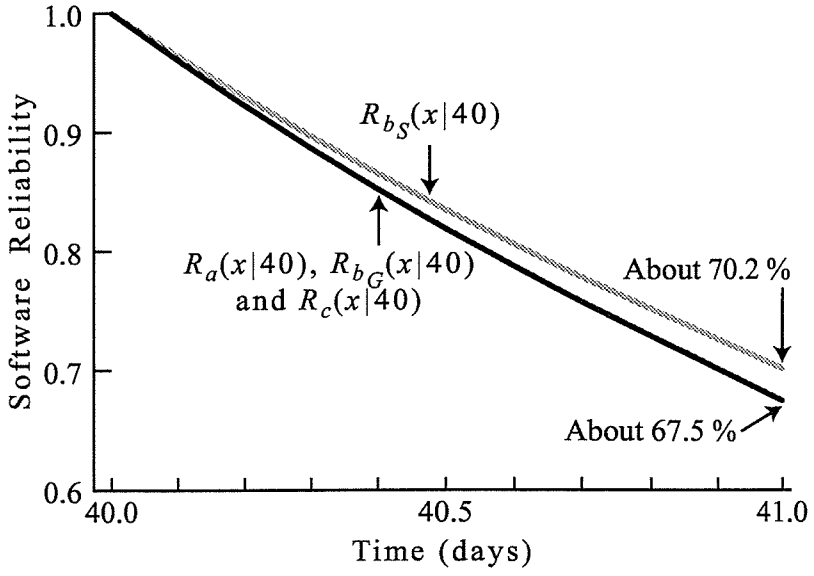

図 4.33: 推定されたソフトウェア信頼度 (DS4).

のユーザ環境下での運用に必要なソフトウェア信頼度を達成するために継続したテストが必要であることが伺える．本評価尺度による結果も，瞬間フォールト発見率の信頼性評価結果と同様，テストケース設計者のテスト習熟性を考慮した簡易習熟性 T-D SRGM を除く SRGM の推定値が厳しい値であることが確認できる．

さらに，式 (3.21) で表される瞬間 MTBF をそれぞれの SRGM により推定した結果を図 4.34 に示す．図 4.34 から，現時点 ($t = 40$) における瞬間 MTBF の推定値はそれぞれ，

$$MTBF_{\mathrm{I}\,a}(40) = MTBF_{\mathrm{I}\,b_G}(40) = MTBF_{\mathrm{I}\,c}(40) = 2.47\ (\text{日}),$$
$$MTBF_{\mathrm{I}\,b_S}(40) = 2.72\ (\text{日}),$$

であることがわかる．この結果は，前述の瞬間フォールト発見率およびソフトウェア

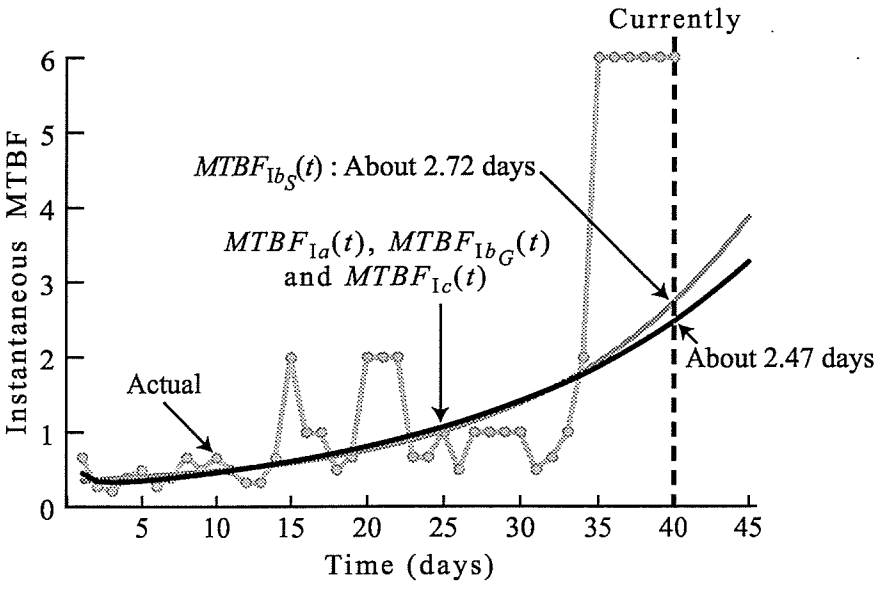

図 4.34: 推定された瞬間 MTBF (DS4).

信頼度による信頼性評価結果と同様，テストケース設計者のテスト習熟性を考慮した簡易習熟性T-D SRGM以外のSRGMの推定値が厳しい値であることが確認できる．また，現時点において同一環境下でテストを継続することにより，瞬間MTBF値が急速に向上する様子も伺える．

最後に，DS4に対するテスト管理施策として，殆どの信頼性評価尺度で厳しい値を示す基本形T-D SRGM，一般習熟性T-D SRGMおよび不完全デバッグT-D SRGMを用いた定量的信頼性評価結果より，

- 瞬間フォールト増加率 ＜ 瞬間フォールト発見率
  瞬間フォールト発見率 ＜ 0.4 (件/日) で高レベル
- 1日後のソフトウェア信頼度は約67.5 %以上
  ユーザ環境下での運用を考慮した場合でも約67.1 %
- 瞬間MTBFは約2.47 (日)

であることがわかる．また，残存フォールト数が約7件であること，および上記信頼性評価結果から，継続したテストが必要であると判断できる．しかし，テスト空間占有率もソフトウェアシステム全体に拡大しており，テストにより発見可能となる瞬間フォールト増加率も0 (件/日) であることから，今後ソフトウェア信頼度および瞬間MTBFを向上させるために，ユーザ運用を想定したテストケースを新規追加設計し，それらのテストケースを実施することなどが今後のテスト施策として考えられる．

4.4 適合性評価基準

ソフトウェア開発のテスト工程におけるフォールト発見事象を，どのSRGMを用いて記述し，ソフトウェアの品質・信頼性評価を実施するのかは，開発管理者にとって非常に興味深い問題である．この問題を経験や勘だけで議論することは，非常に曖昧で危険な行為である．そこで，適切なSRGMを選択する方法として，実測フォールトデータに対する適合性を評価するための基準について議論する．

本節では，4.1節で求めた信頼度成長パラメータの最ゆう推定値を用いて，実測フォールトデータDS1～DS4に対して，第3章で議論した4種類のテスト空間依存型SRGMと，既存のSRGMとの適合性比較を実施する．採用した既存モデルは，テスト空間を考慮したこれらのモデルと同様，NHPPに基づくモデルであり，NHPPモデルを代表する指数形SRGM[12]および遅延S字形SRGM[13]である．また，適合性比較を実施するための基準として，

- 平均偏差2乗和 (average sum of square-differences)
- 対数ゆう度関数値 (log-maximum likelihood function)
- 赤池情報量規準 (Akaike information criterion)

の3種類を取り上げる．

4.4.1　平均偏差2乗和 [3]–[6]

平均偏差2乗和は，実測フォールトデータと推定値との誤差を評価するもので，観測データ数を n とし，推定された平均値関数 $\widehat{H}(t)$ とすると，

$$SSE = \frac{1}{n}\sum_{k=1}^{n}\left[y_k - \widehat{H}(t_k)\right]^2, \tag{4.1}$$

により算出することができ，その値が小さいほど実測フォールトデータに対する適合性の良さを意味する．

6種類の比較対象SRGMの平均偏差2乗和に基づく比較結果を表4.5に示す．この平均偏差2乗和を用いた比較結果から，DS1に対しては一般習熟性T-D SRGM，およびDS2～DS4に対しては不完全デバッグT-D SRGMが他のSRGMと比較して良い適合性を示す結果が得られた．また，全般的に既存モデルよりもテスト空間を考慮したSRGMの方が良い適合性を示すことが確認できた．

表 4.5: 平均偏差2乗和に基づいた比較結果．

Data Set	$H_a(t)$	$H_{b_S}(t)$	$H_{b_G}(t)$	$H_c(t)$	Exponential SRGM	Delayed S-shaped SRGM
DS1	21.150	15.963	15.961	21.224	58.234	21.295
DS2	13.862	14.313	13.653	13.394	29.100	18.449
DS3	3.2963	2.4093	2.4092	1.6410	3.9631	6.0110
DS4	2.1516	2.1952	2.1558	2.1515	2.2276	8.8098

4.4.2 対数ゆう度関数値 [3]–[6]

対数ゆう度関数値は，実測フォールトデータに対するゆう度関数，つまり式(3.15)の自然対数をとった

$$\ln L = \sum_{k=1}^{n} (y_k - y_{k-1}) \cdot \ln \left[H(t_k) - H(t_{k-1}) \right] - \sum_{k=1}^{n} \ln \left[(y_k - y_{k-1})! \right] - H(t_n), \tag{4.2}$$

の値であるから，その値が大きいほど実測フォールトデータに対する適合性の良さを意味する．

6種類の比較対象SRGMの対数ゆう度関数値に基づく比較結果を表4.6に示す．この対数ゆう度関数値を用いた比較結果から，DS1に対しては簡易習熟性T-D SRGMおよび一般習熟性T-D SRGM，DS2およびDS4に対しては一般習熟性T-D SRGM，およびDS3に対しては不完全デバッグT-D SRGMが他のSRGMと比較して良い適合性を示す結果が得られた．また，本評価基準においても，全般的に既存モデルよりもテスト空間を考慮したSRGMの方が良い適合性を示すことが確認できた．

表 4.6: 対数ゆう度関数値に基づいた比較結果．

Data Set	$H_a(t)$	$H_{b_S}(t)$	$H_{b_G}(t)$	$H_c(t)$	Exponential SRGM	Delayed S-shaped SRGM
DS1	-89.775	-84.825	-84.825	-89.778	-90.950	-89.775
DS2	-69.923	-70.036	-69.784	-69.817	-74.809	-71.304
DS3	-49.424	-48.931	-48.931	-48.371	-50.990	-50.881
DS4	-54.844	-55.788	-54.724	-54.844	-56.017	-57.683

4.4.3　赤池情報量規準 [30]–[32]

前節で，モデルの良し悪しを評価する基準の１つとして，対数ゆう度関数値について議論した．しかし，一般的に対数ゆう度関数値は，本当の値 (期待値) に比べて大きくなり易いという偏りをもつ．この傾向はモデルの自由パラメータ数が大きいほど著しい．これは，対数ゆう度関数値を用いた比較によってモデル選択する場合，自由パラメータ数の多いモデルほど選ばれ易いことを示している．

対数ゆう度関数値の期待値に対する偏りの程度とモデルの自由パラメータ数の間の関係は

$$(\text{モデルの最大対数ゆう度}) \ - \ (\text{モデルの自由パラメータ数}),$$

が近似的に期待値の不偏推定量として導かれることから，赤池情報量規準 (以降，AIC と略す) は，

$$AIC = -2 \times (\text{モデルの最大対数ゆう度関数値}) + 2 \times (\text{モデルの自由パラメータ数}), \tag{4.3}$$

により与えられ，この値自体の大小よりは，AIC の値の差の大小に意味がある．つまり，適用された各モデルの AIC の値を求めて，その差が１～２程度以上であるなら，AIC の値の差は有意と考えられ，値の小さい方のモデルの適合性が良いと言える．しかし，AIC の値の差が１よりも小さい場合は，これらのモデルの優劣の判断はできず，どちらも同程度であることを意味する．

６種類の比較対象 SRGM の AIC に基づく比較結果を表 4.7 に示す．この AIC を用いた比較結果から，DS1 に対しては簡易習熟性 T-D SRGM，DS2 および DS4 に

表 4.7: AIC に基づいた比較結果.

Data Set	$H_a(t)$	$H_{b_S}(t)$	$H_{b_G}(t)$	$H_c(t)$	Exponen-tial SRGM	Delayed S-shaped SRGM
DS1	185.55	177.65	179.65	187.56	185.90	183.55
DS2	145.85	148.07	149.57	147.63	153.62	146.61
DS3	104.85	105.86	107.86	104.74	105.98	105.76
DS4	115.69	119.58	119.45	117.69	116.03	119.37

対しては基本形 T-D SRGM，および DS3 に対しては不完全デバッグ T-D SRGM が他の SRGM と比較して良い適合性を示す結果が得られた．また，前述の評価基準と同様に本評価基準においても，全般的に既存モデルよりもテスト空間を考慮した SRGM の方が良い適合性を示すことが確認できた.

4.5 比較結果の考察

前節で議論した適合性比較について，その比較結果を各評価基準のもつ特徴を踏まえて考察する.

平均偏差 2 乗和および対数ゆう度関数値は，モデルの推定値が，単純に実測フォールトデータに対してどれだけ近い値を推定しているかによって適合性の優劣を判断するものである．したがって，これらの評価基準から良い結果を得られたモデルは，実際のソフトウェア開発におけるテスト工程のフォールト発見事象を比較対象モデルより忠実に表現したモデルであると考えられる．ここで，表 4.5 および表 4.6 の比較結果からも明らかなように，テスト空間を考慮した SRGM の方が既存モデルより良い結果を得ている.

特に，表4.5から，フォールトデータがS字形成長曲線としてプロットされるデータセットDS1において，テストケース設計者のテスト習熟性を考慮したテスト空間の一般形，つまり一般習熟性T-D SRGM，およびフォールトデータが指数形成長曲線としてプロットされるデータセットDS3においては，フォールト修正時において新規フォールトが作り込まれる可能性を考慮した不完全デバッグT-D SRGMの適合性が最も良いことがわかる．さらに，表4.6から，フォールトデータがS字形成長曲線としてプロットされるデータセットDS1において，テストケース設計者のテスト習熟性を考慮した習熟性T-D関数に基づくSRGMの適合性が最も良いことがわかる．また，表4.5および表4.6から，DS2およびDS4においては，4種類のテスト空間依存型SRGM共に同程度であるため，モデルパラメータの少ない基本形T-D SRGMによる定量的信頼性評価を実施しても問題無いことが伺える．

AICは，対数ゆう度関数値に加えて，モデルパラメータの数も考慮した評価基準である．つまり，対数ゆう度関数値に著しい差が見られないときは，できるだけモデルパラメータの数が少ないモデルが良いと考えている．そこで，表4.7から，フォールトデータがS字形成長曲線としてプロットされるデータセットDS1において，テストケース設計者のテスト習熟性を考慮したテスト空間の簡易形，つまり簡易習熟性T-D SRGMのAICと，既存の最適モデル(遅延S字形SRGM)のAICとの差が**5.9**，またフォールトデータが指数形成長曲線としてプロットされるデータセットDS3において，新規フォールト混入可能性を考慮した不完全デバッグT-D SRGMのAICと，既存の最適モデル(遅延S字形SRGM)のAICとの差が**1.02**となり，テスト空間を考慮したこれらのモデルの適合性の良さがわかる．他のデータセットDS2およびDS4においては，AICの値が**1**以下となることから，比較対象モデル間でも

優劣の判断はできず，どれも同程度であることを意味する．これは，前述の２つの評価基準と同一の比較結果が得られることから，お互いの適合性評価を裏付ける結果であることが確認できる．AICによる評価基準であまり良い結果が得られなかったのは，テスト空間依存型SRGMにおいて，フォールト発見事象を厳密に捉えることを目的として，既存のSRGMよりもモデルパラメータが増えたことで，モデルの単純性が失われたためと考えられる．

これら３種類の評価基準の結果から総合的に判断すれば，提案した４種類のテスト空間依存型SRGMは，従来モデルよりもモデルパラメータの数が増えたことにより，パラメータ推定が煩雑になるが，実測フォールトデータに対する適合性という観点では向上しており，テスト工程におけるフォールト発見事象を記述するのに十分有効であると考えられる．これらのモデルは，指数形成長曲線およびＳ字形成長曲線を示す両タイプのフォールトデータに適用できる柔軟性の高いモデルであることが伺える．特に，指数形成長曲線としてプロットされるフォールトデータには不完全デバッグT-D関数に基づくSRGM，およびＳ字形成長曲線としてプロットされるフォールトデータには習熟性T-D関数に基づくSRGMの適合性が良いといえる．

4.6 まとめ

本章では，第３章で提案した４種類のテスト空間依存型SRGMを，実際のソフトウェア開発現場で採取された４種類のデータセットを用いて，最ゆう法による各モデルパラメータの推定を実施した．さらに，これらのSRGMから導出される定量的な信頼性評価尺度の推定結果の考察を行い，各データセット毎に今後実施し得るテスト施策について議論した．特に，テスト空間占有率の時間的推移に影響を及ぼす

テスト要因を反映したパラメータの推定結果から，テスト対象ソフトウェアシステム内の部品化されたモジュールの再利用率，およびテストケース設計者のテスト習熟性についても議論した．

また，3種類の適合性比較基準を採用して，既存のNHPPモデルとの適合性比較を実施した．これらの比較結果から，提案した4種類のテスト空間依存型SRGMは，テスト時間に対する累積フォールト数の傾向が指数形成長曲線およびS字形成長曲線を示す両タイプのフォールトデータに適用できる柔軟性の高いモデルであることが確認できた．そして，指数形成長曲線としてプロットされるフォールトデータには不完全デバッグT-D SRGM，およびS字形成長曲線としてプロットされるフォールトデータには習熟性T-D SRGMの適合性の良さが確認できた．

以上のことから，適合性の良いSRGMは，第3章で議論した5種類の定量的信頼性評価尺度により厳しい推定値を示すことが確認できた．ゆえに，最適なSRGMを用いてソフトウェアシステムの品質・信頼性評価を実施することが重要であり，これらの信頼性評価尺度を組み合わせたテスト管理施策を実施することで，効率的かつ経済的に高品質・信頼性ソフトウェアシステムの開発が可能になることを示唆している．

第5章　ソフトウェアテスト管理ツールの構築

一般にテスト工程は，ソフトウェア製品出荷後の品質および信頼性をユーザに対して保証するために大変重要な工程である．事実，ソフトウェア開発に費やされる総開発労力の半分以上が，テスト工程で費やされているのが実情である．このような観点から，テスト工程においてソフトウェアシステムに潜在する総フォールト数およびそれに関連する信頼性尺度を高精度で予測することが可能であれば，ソフトウェアシステムの品質および信頼性を定量的に評価することが可能になる．さらに，テスト進捗度を評価すること，およびソフトウェア製品の最適リリース (出荷) 時間を予測することが可能になる．

近年，多種・多様化するソフトウェアシステムを効率的かつ経済的に開発するために，モジュールあるいは機能単位でパッケージ化することにより，高信頼性ソフトウェアシステムの短期開発を実現している．しかし，パッケージ化された同一のモジュールおよび機能 (以降，共通化モジュールと略す) を複数のソフトウェアシステムに使用するため，共通化モジュールにおいて，プログラム内の人為的誤りや欠陥であるフォールトが発生した場合，共通化モジュールを組み込む全ソフトウェアシステムおよび開発中のプロジェクトにおいて，同一現象の発生の有無を確認する必要がある．さらに，情報伝達不足による問題が様々な形で発生する．また，ソフ

表 5.1: 品質・信頼性評価ツールの一例.

ツール名	組み込みモデル
SORPS[22]	・指数形 SRGM ・遅延 S 字形 SRGM ・習熟 S 字形 SRGM
SPARC[23]	・遅延 S 字形 SRGM ・ロジスティック曲線モデル ・ゴンペルツ曲線モデル
ソフトウェア信頼性評価プログラム [24]	・指数形 SRGM ・遅延 S 字形 SRGM ・習熟 S 字形 SRGM ・ロジスティック曲線モデル ・ゴンペルツ曲線モデル
SOREM[25]	・指数形 SRGM ・遅延 S 字形 SRGM ・ロジスティック曲線モデル ・ゴンペルツ曲線モデル
SRET[26],[27]	・指数形 SRGM ・遅延 S 字形 SRGM ・習熟 S 字形 SRGM ・ロジスティック曲線モデル ・ゴンペルツ曲線モデル

トウェア開発を支援する多数の CASE ツールが開発されているが，テスト工程において観測されたフォールトデータを分析し，ソフトウェアシステムの品質・信頼性およびテスト進捗度を定量的に評価することが可能な実用的ツールはあまり無く，該当するツールが存在したとしても高度な専門的知識を必要とするため，テスト管理

者による使用は非常に困難である．ここで，SRGM を組み込んでソフトウェアの品質・信頼性評価が実行できる代表的な CASE ツールの一例を表 5.1 に示す．

本章では，実際のソフトウェア開発現場での品質・信頼性管理の取り組みとして構築したソフトウェアテスト管理 (software and firmware testing-management，以下 SAFEMAN と略す) ツール，およびデータベースによるフォールト情報の一元管理，かつ全プロジェクトにおいて情報共有および認識が可能な新管理システムについて議論する．そこで，5.1 節では現在の開発プロジェクトの運用方法における問題点抽出を行う．そして，5.2 節ではその問題点の改善方策を議論する．最後に，5.3 節では一般的なテスト管理者および実務者が使用可能な CASE ツールとして開発した SAFEMAN ツールおよび新管理システムの効果について議論する．さらに，両者を連携させることにより得られた効果についても記述する．

5.1 現在の問題点

一般的なソフトウェア開発のテスト工程において，テスト管理者は各開発プロジェクト毎に，以下のような項目を主に管理している：

- ソフトウェア品質・信頼性管理
- テスト進捗度管理
- フォールト情報管理

これらの管理情報において，効率的かつ経済的なテストを実施するため，以下に示す問題点抽出を行った．

5.1.1　ソフトウェア品質・信頼性管理

現在，テスト工程において，ソフトウェア品質・信頼性の管理および定量的評価は，NHPPに基づく2つの基本的なSRGM，つまり指数形SRGM[12]および遅延S字形SRGM[13]や決定論的な回帰モデルであるゴンペルツ曲線モデルから導出される定量的な評価尺度を用いて実施している[5],[6].

このとき，テスト管理者にとって，担当プロジェクトのテスト工程におけるフォールト発見事象を，どのSRGMを用いて記述し，ソフトウェア品質・信頼性の定量的評価を行うかということは，非常に興味深い問題である．しかし，この問題を経験や勘だけで議論するのは，非常に曖昧で極めて危険な行為である．つまり，専門的知識に裏付けされた情報を的確に判断し，最も適合性のよいSRGMによるソフトウェア品質・信頼性の定量的評価が必要である．

5.1.2　テスト進捗度管理

テスト管理者にとって，ソフトウェアシステムの品質・信頼性の定量的評価も重要であるが，テスト進捗度の良好さやテスト工程の安定性を管理することも重要課題である．

しかし現在は，予め設計したテストケースの総数および現在までの投入量により，テスト進捗度を管理している．これは，テストケース消化率であり，実際のテスト進捗度を表すものではない．なぜなら，低品質のテストケースを投入した場合，テストケースにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体の範囲は狭く，形成されるテスト空間も小さい．ゆえに，フォールトの存在が確認できない場合も考えられる(図2.3参照)．また，テスト工

程後期あるいは実際の運用・保守段階で発見されたフォールトを修正および除去する場合，テスト工程初期と比較して新規フォールトが作り込まれる可能性が増大し，ソフトウェアシステムと共にテスト空間が拡大し続ける場合も考えられる (図 2.5 参照)．つまり，現在のテスト進捗度管理では，ソフトウェア品質・信頼性はテストケースの品質およびデバッグ作業者のテスト習熟性に左右されており，テストが不十分な状態で出荷される場合もあると考えられる．

5.1.3 フォールト情報管理

現在，様々なドキュメントをイントラネット上に構築したデータベースおよびファイルサーバを用いて情報の共有化を行っている．また，テスト工程において発見されたフォールト情報も，データベースに登録することにより，全プロジェクト要員が参照可能な状態になっている．

しかし，フォールト情報は各プロジェクト毎に管理されているため，積極的に他プロジェクト情報を参照していないのが現状である．したがって，共通化モジュールにおいてフォールトが発見された場合，共通化モジュールを使用する全プロジェクトへの通知が必要になるが，情報伝達不足による**フォールト修正漏れ**問題が発生し得る．ゆえに，同時進行する複数プロジェクト間で密接な連携が必要である．

5.2 改善方策

前節で述べた問題点を解決するために，第 3 章で議論したテスト空間関数に基づく SRGM を組み込んだ SAFEMAN ツールを，要求仕様定義－設計－コーディング－テストの手順に従って開発する．そこで，システムを再構築するために，上流工程

である要求仕様定義を以下に示す：

(1) 専門的知識を持たない一般管理者が使用可能なCASEツールの開発

- フォールトデータの詳細な分析過程を知らなくても，容易に現状が把握できる画面構成
- フォールト発見事象を記述するSRGMの自動選択
- 各評価尺度において収集可能な情報のレポート化

(2) フォールト情報入力者が意識することなく，関係者へフォールト情報を通知

(3) マウスおよびキーボードによるフォールトデータの登録，未知パラメータの推定，推定モデルに対する適合度検定，およびフォールトデータと推定結果のグラフ化などの幾つかのモジュール(クラス)で構成(図5.1参照)

(4) 他のデータベースと連携するためにMicrosoft Excel®のCSVフォーマットファイルによるフォールトデータのインポート/エキスポートをサポート

(5) パーソナルコンピュータ市場で殆どのシェアを占めるWindows95®，Windows98®およびWindowsNT®のOS上で動作するスタンドアローンアプリケーションとして開発

ここで，旧管理システムから新管理システム導入までの変遷を図5.2に示す．以下に新管理システムとして組み込んだ機能，つまりSAFEMANツールに組み込んだ定量的評価尺度の詳細とデータベースおよびファイルサーバの運用方法改善について記述する．

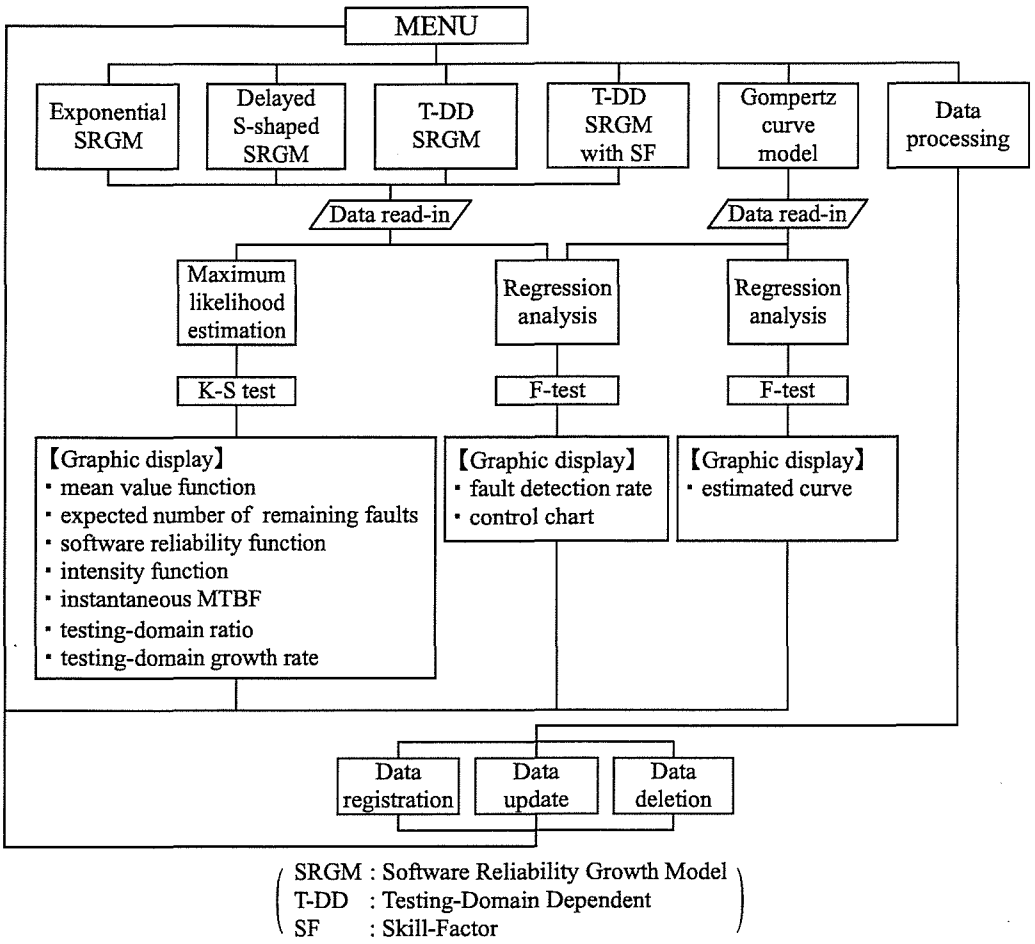

図 5.1: SAFEMAN ツールの構成図.

5.2.1 SAFEMAN ツールの開発

SAFEMAN ツールでは，NHPP に基づく２つの基本モデル，および２つの拡張モデルを採用する．２つの基本モデルとは，テスト時間に対する累積フォールト数の傾向が指数形成長曲線およびＳ字形成長曲線としてプロットされるフォールト発見事象を記述するための指数形 SRGM[12] および遅延Ｓ字形 SRGM[13] である．２つの拡張モデルとは，指数形成長曲線およびＳ字形成長曲線の両タイプが記述可能な

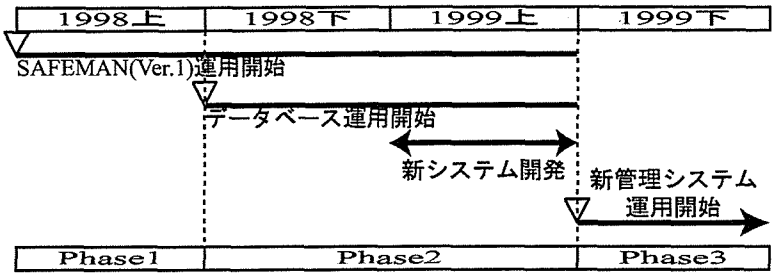

図 5.2: 管理システムの変遷.

テスト空間に基づく SRGM である.

また，決定論的な回帰モデルであるゴンペルツ曲線モデルも採用する．ゴンペルツ曲線モデルは，傾向曲線の収束値をソフトウェアシステム内に潜在する総フォールト数として回帰分析により推定することが目的である．つまり，テスト時刻 t と発見された累積フォールト数 $N(t)$ の関係に確率則を仮定しない決定論的モデルである.

さらに，本ツールでは，NHPP モデルにより導出可能な信頼性評価尺度として，

(1) 期待残存フォールト数

(2) ソフトウェア信頼度

(3) 瞬間 MTBF

(4) テスト空間の定量化

(5) テスト進捗度管理図

を採用する．上記評価尺度 (1) ～ (3) においては，既に 3.4 節で議論しているので省略する．ここでは，まだ議論していない評価尺度 (4) および (5) について記述する.

テスト空間の定量化 [15]

テスト空間とは，第２章で議論したように，投入したテストケースにより影響を受けるソフトウェアシステム内のモジュールおよび機能のテストパスに関する集合体のことである．つまり，テスト空間の定量的な表現が可能になれば，テスト管理者にとって重要な管理項目の１つである，テスト対象ソフトウェアシステム内のテストパス網羅率が擬似的に把握することができる．これは，テストパス網羅率が100%に未達である場合，幾つかのモジュールおよび機能において，テスト未実施あるいは不十分な状態であると考えられる．しかし，ブラックボックス・テスト技法が用いられるシステムテストにおいて，大規模ソフトウェアシステムのテストパス網羅率を計測することは不可能である．

そこで，大規模なソフトウェア開発のシステムテストにおいて，ソフトウェアシステム内のテスト空間占有率と総発見フォールト数の関係，および単位時間当りに投入されるテストケースによるテスト空間の拡大率とテストによりテストケース設計者が発見可能となるフォールトの増加数の関係について，定量的に評価を行った上で，これらの可視化を行う．これは，式(2.2)–式(2.12)により前者の関係，および式(2.15)–式(2.18)により後者の関係の時間的挙動をグラフ化することにより可能となる．概念図は，図2.1，図2.3，および図2.5に既に示した．ゆえに，テスト空間を定量化することにより，実測フォールトデータからソフトウェアシステム内におけるテスト空間占有率の統計的推定が可能となり，従来では計測不可能であった大規模ソフトウェアシステムのテストパス網羅率の推定，つまり擬似的なカバレッジ計測が可能となる．

テスト進捗度管理図 [6],[28]

テスト管理者は，ソフトウェア品質・信頼性の定量的評価のみならず，テスト進捗度の良好さやテスト工程の安定性を判断する必要がある．ここでは，統計的品質管理の一手法である管理図法を用いることにより，ソフトウェアのテスト進捗度管理の統計的方法について議論する．本方法は，SRGMから導出される強度関数 $h(t)$，つまり瞬間フォールト発見率に基づくものである．

ここでは，遅延S字形SRGM[13]に適用した場合を中心に議論する．遅延S字形SRGMの強度関数より，次の関係式を得る：

$$\ln[Z_s(t)] \equiv \ln\left[\frac{h_s(t)}{t}\right] = \ln a + 2\ln b - bt. \tag{5.1}$$

式(5.1)は，平均瞬間フォールト発見率 $Z_s(t)$ の対数値とテストの経過時間との関係が，線形性を有する事を意味し，テストが良好に進行し，テストにより安定的に信頼度成長が確保されているとき，すなわちテスト工程が管理状態にあるならば，平均瞬間フォールト発見率の対数値がテスト時間の経過と共に直線的に減少していくことを示唆している．

式(5.1)から，モデルパラメータ a および b を最小2乗法により推定することができ，観測されたフォールトデータを用いて適合性を回帰分析により確認できる[3],[29]．モデルパラメータの推定値を，$\hat{a}$ および $\hat{b}$ とすると，$Y\,(=\ln Z_s(t))$ は

$$\begin{aligned}\hat{Y} = \ln\widehat{Z_s}(t) &= \ln\hat{a} + 2\ln\hat{b} - \hat{b}t \\ &= \overline{Y} - \hat{b}(t-\bar{t}),\end{aligned} \tag{5.2}$$

により推定できる．ここで，観測されたフォールトデータが $(t_k,\ Z_k)\,(k=1,2,\cdots,n)$

であると仮定し，

$$\overline{Y} = \frac{1}{n}\sum_{k=1}^{n} Y_k, \qquad Y_k = \ln Z_k, \qquad \bar{t} = \frac{1}{n}\sum_{k=1}^{n} t_k, \tag{5.3}$$

とする．式(5.3)において，t_kは一定のテスト時刻$(0 < t_1 < t_2 < \cdots < t_n)$，$Z_k$はテスト時刻$t_k$における平均瞬間フォールト発見率$Z_s(t)$の観測値を表す．このとき，式(5.2)の説明変数$t$に対する回帰関係で説明される変動，および誤差変動は，それぞれ

$$S_b = \sum_{k=1}^{n}\left(\widehat{Y_k} - \overline{Y}\right)^2 = \widehat{b}^2\sum_{k=1}^{n}(t_k - \bar{t})^2, \qquad S_e = \sum_{k=1}^{n}\left(Y_k - \widehat{Y_k}\right)^2, \tag{5.4}$$

となる．したがって，式(5.4)に対する不偏分散は，それぞれ

$$V_b = S_b, \qquad V_e = \frac{S_e}{n-2}, \tag{5.5}$$

であるから，これを用いて分散分析を行えば，式(5.2)の回帰式の観測データに対する適合性，すなわちモデルの適合性を統計的に確認することができる．

式(5.2)において，$t = t_0\,(t_0 \geq t)$における平均瞬間フォールト発見率の対数値$Y_0 = \ln Z_s(t_0)$の推定値$\widehat{Y_0}$に対する$100(1-\eta)\,\%$信頼区間は，

$$\widehat{Y_0} \pm t\left(n-2, 1-\frac{\eta}{2}\right)\sqrt{\mathrm{Var}\left[\widehat{Y_0}\right]},$$

$$\mathrm{Var}\left[\widehat{Y_0}\right] = \left\{1 + \frac{1}{n} + \frac{(t_0 - \bar{t})^2}{\sum_{k=1}^{n}(t_k - \bar{t})^2}\right\} V_e, \tag{5.6}$$

により与えられる．ここで，$\mathrm{Var}\left[\widehat{Y_0}\right]$は$\widehat{Y_0}$の分散，$t(k,\ m)$は自由度$k$のt分布の$100m\%$点である．

管理図の作成にあたり，式(5.1)の$\ln[Z_s(t)]$を中心線とし，式(5.6)をその管理限界線として用いることにより管理図が作成される．これにより，$\ln[Z_s(t)]$の観測値

を，管理図上にプロットすることにより，そのプロット点が中心線に沿って管理限界内に収まっていれば，テスト進捗度が良好であるとみなし，管理限界外にプロットされたなら，テスト工程に何らかの異常が発生したとみなし，その原因を解析すると共に適切な処置を施す必要がある．

他のSRGMの適用環境においても同様の関係により，テスト進捗度評価のための管理図を作成することができる．

5.2.2　フォールト情報管理

ソフトウェア開発において，他プロジェクトと密接かつスムーズな連携を図るため，以下に示す項目をプロジェクト初期設定としてデータベースへの登録を義務付ける：

- プロジェクト管理者
- リーダー
- 開発するソフトウェアシステムに組み込む共通化モジュール

これは，同時進行中である他プロジェクトのテスト工程において，共通化モジュールでフォールトが発見された場合，フォールト情報をデータベースに登録すると同時に，該当の共通化モジュールを使用する全プロジェクトの管理者およびリーダーにフォールト情報を通知することを目的とする．また，全ドキュメント(仕様書，ソースプログラム，テストケースおよびレビュー記録など)について，各ドキュメント毎にフォールト発生情報あるいは修正・改版を履歴情報として，データベースにおいて記録・管理する運用に改善する．つまり，プロジェクト初期設定登録以前の履歴

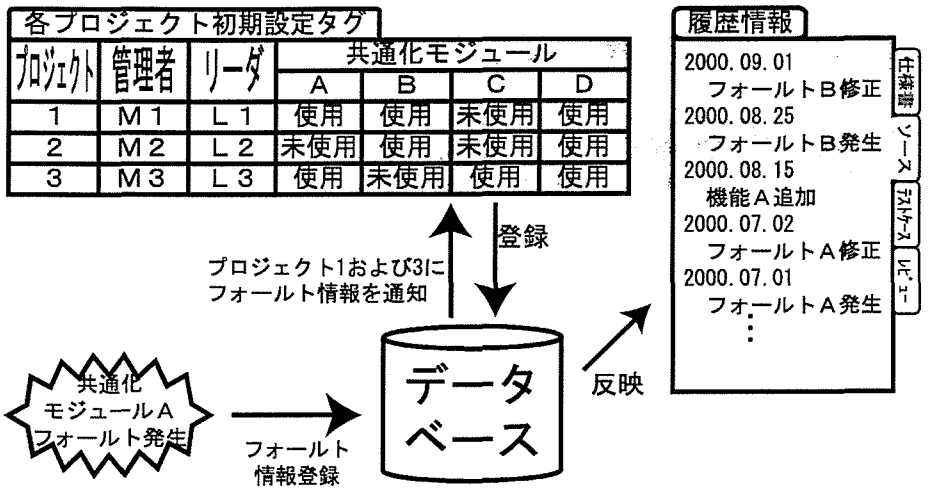

図 5.3: データベースの構成図.

情報が参照可能になることで，各要求仕様書に対する設計・改版時に注意を促せる構成にした (図 5.3 参照).

5.3 新管理システムの効果

前節で議論した改善方策を有する新管理システムについて考察する.

先ず，実際のソフトウェア開発現場におけるテスト工程の手順を以下に示す：

(1) ユニットテスト

(2) モジュールあるいは機能単位での結合テスト

(3) システムテスト 1

(4) システムテスト 2

これらのテスト工程の特徴は，システムテストを 2 工程に分割して，各々のシステム

テストにおいて異なるテストチームが担当するところである．つまり，システムテスト1において，ソフトウェアシステムを開発した部門が，開発者の立場で要求仕様書に基づいてテストケースを設計し，インプリメントした機能の検証を行う．また，システムテスト2において，顧客に対してテスト対象ソフトウェアシステムの品質および信頼性を保証する部門，いわゆる品質保証部門が，使用者の立場で要求仕様書に基づいてテストケースを設計し，実際の顧客運用に対して問題の有無を検証する．ゆえに，各テストチームが独立に，要求仕様書を解釈し，その内容に基づいたテストケースの設計を行っている．これは，各テストチームが異なる観点でテストを行うことにより，テスト漏れ防止およびソフトウェアシステムの品質および信頼性の向上を目的として実施している．

5.3.1　改善効果

次に，システムテスト2の期間中に発見される累積フォールト数およびソフトウェア製品出荷後に顧客先で発生したフォールト数により，新管理システムにおける改善効果の確認を行う．

先ず，システムテスト1において，発見および修正された累積フォールト数のデータを用いて，SAFEMANツールに組み込まれた7つの定量的な評価尺度によるソフトウェアシステムの品質および信頼性評価を実施する．次に，SAFEMANツールにおいて，各評価尺度毎に表示されるテスト対象ソフトウェアシステムの品質・信頼性評価結果および今後の管理施策メッセージに沿ったテスト工程管理を行うと同時に，他プロジェクトで発見されたフォールト情報の反映も行う．図5.4に，SAFEMANツールにより表示された管理施策メッセージ(画面左下の付随情報)および評価尺度

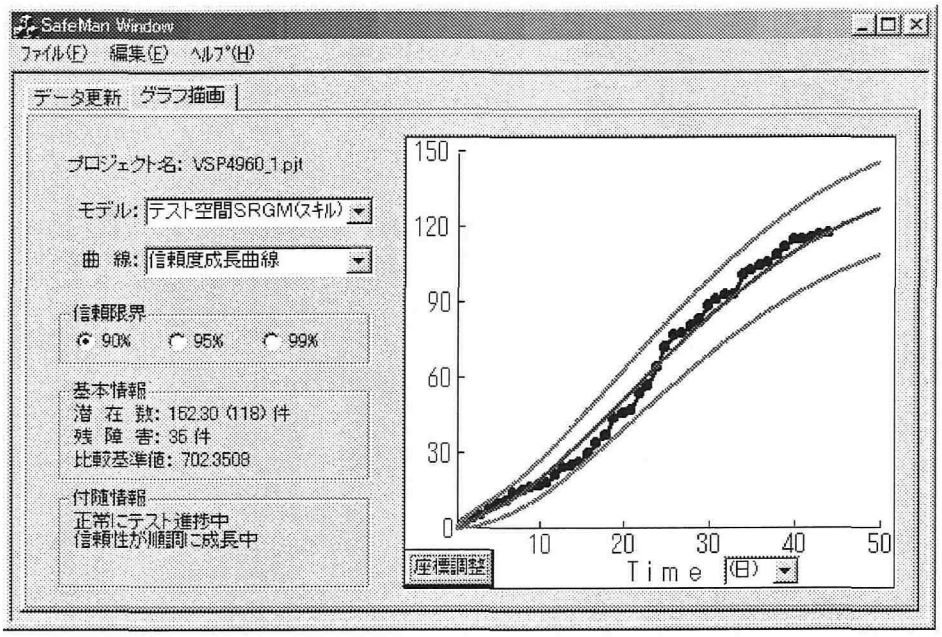

図 5.4: SAFEMAN ツールによる評価尺度および管理施策メッセージ表示の一例.

のグラフ (画面右側) の一例を示す.

上記のテスト工程管理を行った後, 同様にシステムテスト 2 を実施する. このシステムテスト 2 の期間中に発見された累積フォールト数のデータから, 管理システムの改善によるソフトウェアシステムの品質および信頼性の推移を図 5.5 に示す. 図 5.5 は, 1998 年度以前に開発したソフトウェアシステムにおいて, 単位 KLOC(コード行数 $\times 10^3$) 当りの発見フォールト数を 100% とした相対的な割合を表している.

次に, ソフトウェアシステム出荷後 1 年間に, 顧客先で発生したソフトウェア故障数を図 5.6 に示す. 図 5.6 は, 1998 年度以前に出荷したソフトウェアシステムにおいて, 出荷後 1 年間に顧客先で発生したソフトウェア故障数を 100% とした相対的な割合を表している.

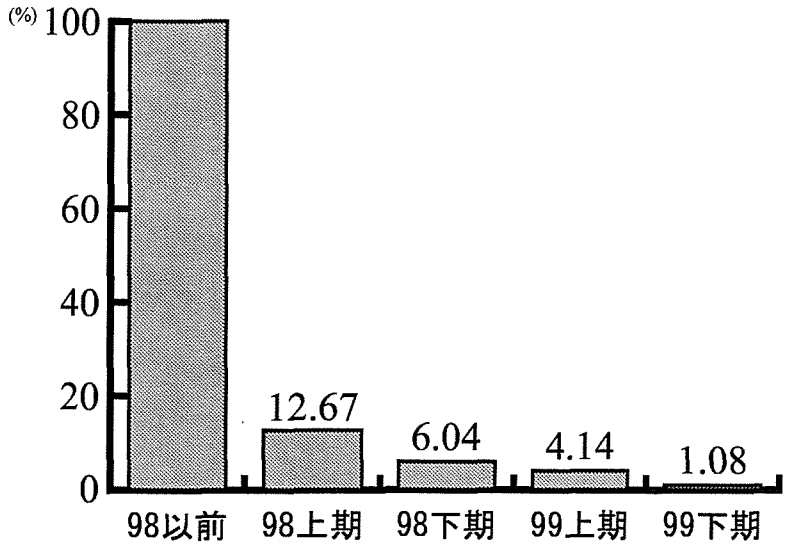

図 5.5: 単位 KLOC 当りの発見フォールト数の改善率.

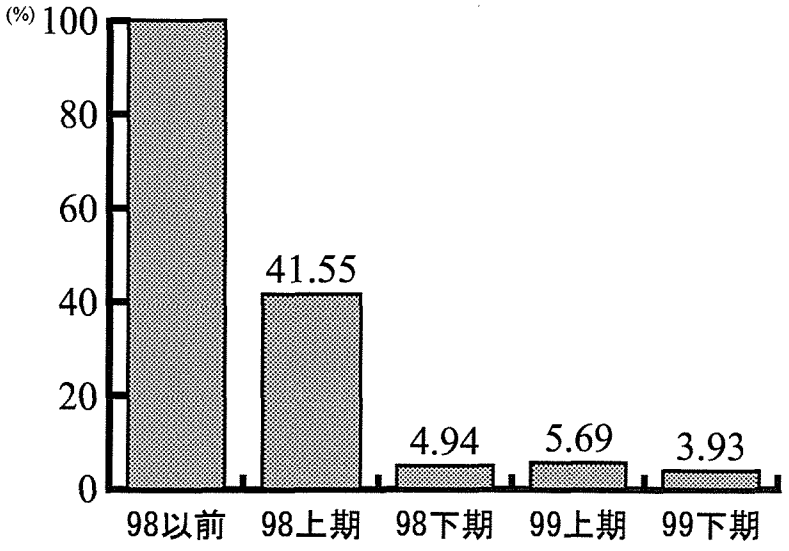

図 5.6: 顧客先で発生したソフトウェア故障数の改善率.

5.3.2 改善効果の考察

データベースおよびファイルサーバによる情報管理，さらにSAFEMANツールによるテスト工程管理，それぞれの手法をソフトウェア開発に採用することで，テスト工程のシステムテスト２の期間中に発見された累積フォールト数は，1998年度以前と比較して約96%削減という結果が得られた．さらに，データベースとファイルサーバによる情報管理，およびSAFEMANツールによるテスト工程管理機能を強化すると共に，両手法を密接に連携させた新管理システムにおいて，システムテスト２の期間中に発見された累積フォールト数は，図5.5から1998年度以前と比較して約99%の削減が実現できた．また，図5.6から，ソフトウェアシステム出荷後，顧客先で発生したソフトウェア故障数について，新管理システムで開発した最初のソフトウェア(99年度下期出荷)製品で約96%の削減に成功し，高品質および高信頼性ソフトウェアシステムの開発が可能になった．

これは，ソフトウェア開発の早期段階から他プロジェクトと連携することにより，他プロジェクトで発生する様々な情報を早急に確認することが可能になり，その情報を反映するか否かの判断が早期に行えるようになったこと，およびテスト工程初期からソフトウェアシステムの品質および信頼性の定量的評価に有益な複数の評価尺度を併用したテスト工程管理が可能になった結果であると考えられる．

5.4 まとめ

本章では，ソフトウェアシステムの品質・信頼性およびテスト進捗度を定量的に評価し，各評価尺度に付随する様々な情報を管理施策メッセージとして表示できるソフトウェアテスト管理ツール“SAFEMAN”と，情報共有化技術を用いたデータ

ベース／ファイルサーバによる情報管理を連携させた新管理システムの開発成果について議論した．

SAFEMANツールには，フォールトデータ分析のために5つのSRGMと，ソフトウェアシステムの品質・信頼性の定量的評価に有益な7つの評価尺度を組み込んだ．また，SAFEMANツールは，ソフトウェア開発のテスト工程において，専門的知識を持たない一般的なテスト管理者でも，フォールト情報をデータベースに登録するだけで，データベースとの連携により手軽に対話形式でソフトウェアシステムの品質・信頼性評価およびテスト進捗度管理が可能であり，プロジェクト管理上極めて有益なツールである．

また，この新管理システムの特徴的な利点と新規性は，フォールトデータの分析結果を視覚的，かつ簡潔に表現することができ，ソフトウェアシステムの品質・信頼性の定量的評価を行うことにより，テスト対象ソフトウェアシステムの現状が容易に把握することができる点にある．そして，データベース／ファイルサーバと連携することにより，他プロジェクトで発生したフォールト情報の早期確認・反映が可能になった．以上のことから，新管理システムの運用以前と比較して，飛躍的な品質・信頼性改善ができたと考えている．

第6章　むすび

6.1　研究の成果

本研究では，ソフトウェア開発の最終工程であるテストにおいて，要求仕様書に基づいてインプリメントされた機能を検証するために投入されるテストケースから影響を受けるテスト空間のソフトウェアシステム内における占有率の時間的推移と，フォールト発見事象を関係づけたNHPPに基づくSRGMを提案した．特に，ソフトウェアシステム内におけるテスト空間占有率，つまりテストにおいて発見されるフォールト数に多大な影響を与えるテスト要因として考えられる

- テストケース設計者のテスト習熟性
- フォールト修正時における新規フォールト混入可能性

に着目してモデル化を行った．また，本研究で提案したこれらのSRGMにより導出された信頼性評価尺度について，テスト工程において観測されたフォールトデータに対する数値例を示した．この結果から，適合性の良いモデルでは，定量的信頼性評価尺度により厳しい推定値を示すことが確認できた．

本研究の目的である既存のSRGMよりも，テスト工程に対する適合性および妥当性がどれだけ向上したかについては，第４章において議論した複数の比較基準を用いたモデル評価を実施することにより検証できた．この適合性評価の結果から，提

案したこれらのSRGMは，既存のSRGMよりもフォールト発見事象を的確に表現しており，実際のテスト工程に対する適合性という観点においては，既存のSRGMより優れていることがわかった．さらに，テスト空間という概念を用いることにより，フォールトデータからソフトウェアシステム内におけるテスト空間占有率の統計的推定が可能となり，従来では不可能であった大規模ソフトウェアシステム内のテストパス網羅率の推定，つまり擬似的カバレッジ計測が可能となった．

最後に，実際のソフトウェア開発現場における信頼性・品質管理の取り組みとして，実際のテスト工程管理に役立つソフトウェアテスト管理(SAFEMAN)ツールの開発，および既存CASEツールを連携させた新管理システム構築の成果について議論した．この新管理システムは，テスト工程において，専門的知識を持たない一般的なテスト管理者でも，フォールトデータの分析結果を視覚的，かつ簡潔に表現することができ，手軽に対話形式でソフトウェア品質・信頼性評価およびテスト進捗管理が可能である．また，データベースおよびファイルサーバを連携させることで，他プロジェクトの発生フォールト情報の早期確認および反映が可能であり，テスト工程管理のみならず，上流工程である設計段階においても有益なシステムであると考える．

6.2　今後の課題

今後の課題として，本研究で提案したこれらのモデルに多数のフォールトデータを適用し，モデルの有効性を検討していく必要がある．さらに，提案した4種類のSRGMにおいて，モデルパラメータが増加したことにより，パラメータ推定が既存のSRGMと比較して煩雑になった．この煩雑さを解消する方策についても検討しな

ければならない．また，提案した4種類のテスト空間依存型SRGMの信頼度成長パラメータである非テスト習熟性pと新規フォールト潜入率βの推定結果において，新規フォールトが作り込まれる状況とテストケース設計者のテスト習熟性の関係についても，詳細に検討していく必要がある．

参考文献

[1] W. A. Wulf: "Programming methodology", *Proc. Symp. High Cost of Software*, pp. 81–98 (1973).

[2] B. W. Boehm, J. R. Brown and M. Lipow: "Quantitative evaluation of software quality", *Proc. 2nd Intern. Conf. Software Engineering*, pp. 529–605 (1976).

[3] 山田茂："ソフトウェア信頼性評価技術"，ＨＢＪ出版局，東京 (1989).

[4] 山田茂，大寺浩志："ソフトウェアの信頼性ー理論と実践的応用"，ソフト・リサーチ・センター，東京 (1990).

[5] 山田茂, 高橋宗雄："ソフトウェアマネジメントモデル入門ーソフトウェア品質の可視化と評価法"，共立出版，東京 (1993).

[6] 山田茂："ソフトウェア信頼性モデルー基礎と応用ー"，日科技連出版社，東京 (1994).

[7] H. Pham : "*Software Reliability*", Springer-Verlag, Singapore (2000).

[8] Z. Jelinski and P. B. Moranda : "Software reliability research", in *Statistical Computer Performance Evaluation*, W. Freiberger (ed.), pp. 465–484, Academic Press, New York (1972).

[9] P. B. Moranda : "Event-altered rate models for general reliability analysis", *IEEE Trans. Reliability*, vol. R-28, no. 5, pp. 376–381 (Dec. 1979).

[10] B. Littlewood : "Theories of software reliability : How good are they and how can they be improved ? ", *IEEE Trans. Software Engineering*, vol. SE-6, no. 5, pp. 489–500 (Sep. 1980).

[11] J. D. Musa, A. Iannino and K. Okumoto : "*Software Reliability : Measurement, Prediction, Application*", McGraw–Hill, New York (1987).

[12] A. L. Goel and K. Okumoto : "Time-dependent error-detection rate model for software reliability and other performance measures", *IEEE Trans. Reliability*, vol. R–28, no. 3, pp. 206–211 (Aug. 1979).

[13] S. Yamada and S. Osaki : "Software reliability growth modeling : Models and applications", *IEEE Trans. Software Engineering*, vol. SE-11, no. 12, pp. 1431–1437 (Dec. 1985).

[14] 山田茂，半谷知久，尾崎俊治："一般化された遅延Ｓ字形ソフトウェア信頼度成長モデルとその適合性評価に関する考察"，電子情報通信学会論文誌，vol. J76–D–I, no. 11, pp. 613–620 (Nov. 1993).

[15] 大寺浩志, 山田茂, 成久洋之 : “テスト空間を考慮したソフトウェア信頼度成長モデル”, 電子情報通信学会論文誌, vol. J73–D–I, no. 2, pp. 170–174 (Feb. 1990).

[16] S. Yamada, H. Ohtera and M. Ohba: “Testing-domain dependent software reliability models”, *Computers & Mathematics with Applications*, vol. 24, no. 1/2, pp. 79–86 (Jul. 1992).

[17] 山田茂 : “潜入フォールトによる不完全デバッグを考慮したソフトウェア信頼度成長モデル”, 電子情報通信学会論文誌, vol. J80–A, no. 2, pp. 363–370 (Feb. 1997).

[18] S. Yamada, K. Tokunou and S. Osaki : “Imperfect debugging models with fault introduction rate for software reliability assessment”, *Intern. J. Systems Science*, vol. 23, no. 12, pp. 2241–2252 (Dec. 1992).

[19] A. O. Allen : “*Probability, Statistics and Queueing Theory*”, Academic Press, New York (1978).

[20] 日本規格協会編 : “統計数値表－ JSA”, 日本規格協会, 東京 (1972).

[21] 山本美保, 並河安則, 山田茂 : “ソフトウェア信頼性評価ツールの導入～ソフトウェア信頼度成長モデルの適用と評価～”, 第 18 回ソフトウェア生産における品質管理シンポジウム発表報文集, pp. 201–208 (1998).

[22] M. Ohba : “Software reliability analysis models”, *IBM J. Research and Development*, vol. 28, no. 4, pp. 428–443 (Jul. 1984).

[23] 中村英夫，内平直志，佐藤誠，水谷博之："ソフトウェアの品質をテスト工程に入る前に予測する手法"，日経エレクトロニクス，1985年2月25日号，pp. 233–260 (1985).

[24] Y. Komuro : "Evaluation of software products' testing phase–Application to software reliability growth models", in *Reliability Theory and Applications*, S. Osaki and J. Cao (eds.), pp. 188–194, World Scientific, Singapore (1987).

[25] M. Uemura, S. Yamada and K. Fujino : "Software reliability evaluation method : Application of delayed S-shaped NHPP model and other related models", *Proc. Intern. Symp. Reliability and Maintainability*, pp. 467–472 (1990).

[26] 山田茂，磯崎龍史，尾崎俊治："ソフトウェア信頼性評価ツール (SRET) の作成"，電子情報通信学会論文誌，vol. J72–D–I，no. 1，pp. 24–32 (Jan. 1989).

[27] S. Yamada and M. Kimura : "Software reliability assessment tool based on object-oriented analysis and its application", *Annals of Software Engineering*, vol. 8, pp. 223–238 (1999).

[28] 山田茂，杣木秀樹："ソフトウェア信頼度成長モデルによるテスト進捗度評価法"，日本応用数理学会論文誌，vol. 6，no. 4，pp. 317–327 (1996).

[29] 久米均，飯塚悦功："回帰分析"，岩波書店，東京 (1987).

[30] 赤池弘次："情報量規準 AIC とは何か"，数理科学，no. 153，pp. 5–11 (1976).

[31] 坂元慶行，石黒真木夫，北川源四郎："情報量統計学"，共立出版，東京 (1983).

[32] 鈴木義一郎：“情報量規準による統計解析入門”，講談社サイエンティフィク，東京 (1995).

謝　辞

本研究を遂行するにあたり，多大な御指導と御鞭撻を賜った鳥取大学工学部社会開発システム工学科 山田茂教授および河合一教授，知能情報工学科 池原悟教授に深甚な謝意を表します.

本研究を進めるにあたり，様々な面で御支援頂いた鳥取大学工学部社会開発システム工学科 得能貢一助教授をはじめ，山田研究室の方々に深く感謝申し上げます.

平素から，有益な御教授ならびに御指導頂いた法政大学工学部経営工学科 木村光宏助教授に深く感謝申し上げます.

本研究を行うにあたり，並々ならぬ御配慮ならびに御支援頂いた富士通周辺機株式会社 開発統括部の塩谷和則氏ならびに関根浩一氏はじめファームウェア開発グループの方々，多くの方々に感謝申し上げます.

岡山理科大学大学院在学中からの恩師であり，常に励ましの御言葉を頂いた岡山理科大学工学部情報工学科 成久洋之教授に心から感謝申し上げます.

最後に，常日頃から心の支えとなり，励ましてくれた父 良隆，母 昭子，妻 栄子，および息子 崇人ならびに佑丞，娘 志帆に感謝します.

研究業績一覧表

学術論文

1. 藤原隆次，山田茂："テスト習熟性を考慮したソフトウェア信頼度成長モデルとその適合性評価に関する考察"，電子情報通信学会論文誌，vol. J83–A, no. 2, pp. 188–195 (Feb. 2000).

1'. Takaji Fujiwara and Shigeru Yamada : "Software reliability growth modeling based on testing-skill characteristics : Model and application", *Electronics and Communications in Japan*, Part 3, vol. 84, no. 6, pp. 42–49 (2001).

2. Takaji Fujiwara, Shigeru Yamada and Kazunori Shiotani : "A software testing-management tool for reliable software development and its application", *Proc. the Sixth ISSAT International Conference on Reliability and Quality in Design*, pp. 175–179 (Aug. 2000).

3. Takaji Fujiwara and Shigeru Yamada : "A testing-domain dependent software reliability growth model for practical application", *Proc. the Second World Congress for Software Quality*, pp. 821–826 (Sep. 2000).

4. 藤原隆次，関根浩一，塩谷和則，山田茂：“情報共有化による高信頼性ソフトウェア開発の一手法”，プロジェクトマネジメント学会誌，vol. 3，no. 2，pp. 9–14 (Apr. 2001).

5. 藤原隆次，山田茂：“一般化されたテスト空間依存型ソフトウェア信頼度成長モデルとその適合性評価に関する考察”，電子情報通信学会論文誌，vol. J84–A，no. 7，pp. 950–958 (Jul. 2001).

6. Takaji Fujiwara and Shigeru Yamada : “Testing-domain dependent software reliability growth models and their comparisons of goodness-of-fit”, *Proc. the Seventh ISSAT International Conference on Reliability and Quality in Design*, pp. 36–40 (Aug. 2001).

6'. Shigeru Yamada and Takaji Fujiwara : “Testing-domain dependent software reliability growth models and their comparisons of goodness-of-fit”, *International Journal of Reliability, Quality and Safety Engineering*, vol. 8, no. 3, pp. 205–218 (Sep. 2001).

7. 藤原隆次，山田茂：“不完全デバッグ環境を考慮したテスト空間依存型ソフトウェア信頼度成長モデルとその適合性評価に関する考察”，電子情報通信学会論文誌，vol. J85–A，no. 1，pp. 76–83 (Jan. 2002).

研究報告・発表

1. 藤原隆次，大寺浩志，山田茂，成久洋之："テスト空間を考慮したソフトウェアの信頼性評価モデル"，電気関係学会中国支部第40回連合大会講演論文集，122514，p. 238 (Oct. 1989).

2. 藤原隆次，大寺浩志，山田茂，成久洋之："エラーの発見可能性を考慮したソフトウェア信頼度成長モデル"，電子情報通信学会春季全国大会講演論文集，D–238，p. 240 (Mar. 1990).

3. 藤原隆次，大寺浩志，山田茂，成久洋之："テスト空間の時間的推移を考慮したソフトウェア信頼度成長モデル"，電子情報通信学会技術研究報告 (フォールトトレラントシステム研究会)，FTS90–10，pp. 17–22 (Jun. 1990).

4. 藤原隆次，大寺浩志，山田茂，成久洋之："テスト空間の初期拡大率を考慮したソフトウェア信頼性評価モデル"，電気関係学会中国支部第41回連合大会講演論文集，102507，p. 234 (Oct. 1990).

5. 藤原隆次，塩谷和則，山田茂："テスト空間依存型ソフトウェア信頼度成長モデルとその適合性に関する考察"，電子情報通信学会技術研究報告 (ソフトウェア・サイエンス研究会)，SS99–28，pp. 33-40 (Sep. 1999).

6. 藤原隆次，山田茂，関根浩一，塩谷和則：“ソフトウェアテスト管理ツール (SAFEMAN) の開発とその適合性に関する考察”，電子情報通信学会技術研究報告 (信頼性研究会)，R2000–6，pp. 31–36 (Mar. 2000).

7. 藤原隆次，山田茂：“テスト空間依存型ソフトウェア信頼度成長モデルの一般化とその適合性評価に関する考察”，電子情報通信学会技術研究報告 (信頼性研究会)，R2001–8，pp. 41–46 (Mar. 2001).

8. 藤原隆次，山田茂：“不完全デバッグ環境を考慮したテスト空間依存型ソフトウェア信頼度成長モデル”，日本オペレーションズ・リサーチ学会秋季研究発表会，2–D–7，pp. 232–233 (Sep. 2001).

社内報告

1. 藤原隆次：“信頼性解析ツールの開発”，富士通株式会社，高信頼性運動第 30 回全社発表会要旨集，pp. 38-41 (Nov. 1998).

特許

1. 藤原隆次，西村宏光，他：“プリンタ制御方法及びプリンタ装置”，特許庁，特許第 3111248 号 (Sep. 2000).

公開審議中特許

1. 藤原隆次，山田茂："ソフトウェア開発支援装置および記録媒体"，特許庁，特願 2000–239592 (Aug. 2000).

2. 藤原隆次，中山雄二，他："非正立文字の描画方法"，特許庁，特開平 7–77967 (Jul. 1995).

3. 藤原隆次，中山雄二，他:"プリンタ装置"，特許庁，特開平 7–96639 (Apr. 1995).

4. 藤原隆次："階調変換処理方式，そのコンピュータプログラムを記録した記録媒体及びその回路，並びにコンピュータネットワークシステム"，特許庁，特開平 9–205546 (Aug. 1997).

5. 藤原隆次，西村宏光，他："プリント制御方法及びプリンタ装置"，特許庁，特開平 10–161828 (Jun. 1998).

6. 藤原隆次，西村宏光，他："改ページ指示方法，ページ管理方法，プリンタシステム及びプリンタシステム"，特許庁，特開平 11–70712 (Mar. 1999).

7. 藤原隆次，西村宏光，他："環境情報設定方法，印刷装置及び記憶媒体"，特許庁，特開平 11–74897 (Mar. 1999).

8. 藤原隆次，西村宏光，他：“プログラム管理方法及び装置”，特許庁，特開平11–73303 (Mar. 1999).

9. 藤原隆次，西村宏光，他：“同報通信方法，同報通信装置，及び記録媒体”，特許庁，特開 2000–106559 (Apr. 2000).

10. 藤原隆次:“プリンタ，データ処理装置，データ送信装置，印刷制御装置，印刷システム，記録媒体，及び印刷制御方法”，特許庁，特開 2000–103144 (Apr. 2000).

11. 藤原隆次，藤井直人：“印刷装置”，特許庁，特開 2000–211219 (Aug. 2000).

END