# ソフトウェア信頼性評価モデルとその適用に関する研究

中川, 豊

https://doi.org/10.11501/3130975

# 第６章　大規模オンラインシステムの高信頼化試験法

これまでは，開発中あるいは開発されたソフトウエアの信頼性評価の指標である残存フォールト数の推定モデルとその適用法について述べた．本章では，サービスを間近にした時点で，ソフトウエアの信頼性そのものを如何に高めるかということを述べる．ここで述べる試験は，サービス開始前に実施する最終的な試験であり，これまでの試験によって達した信頼性の評価，そして，サービス開始のための信頼性保証の側面も合わせて持っている．

先ず，当該試験のねらいと試験システムの条件を述べ，試験効率化モデルを提示する．次に，その条件を充足するように開発した試験システムの機能と構成について述べ，そのシステムを使用した試験の効果を述べる．すなわち，試験実施時間の短縮効果，試験稼働の削減効果，サービス開始後の平均ダウン時間間隔（ＭＴＢＤ：Mean Time Between Downs）の長さについて，従来と比較すると共に，検出したフォールトの分析結果からフォールト検出に関する有効性を示す．また，それらのフォールトの検出難易度による分類結果と信頼性の関係を述べる．

## ６．１　大規模システムの総合試験の課題

大規模システムを構成するプログラム個々の品質（信頼性，性能，等）がよくても，それらを統合してひとつのシステムとして組み上げたとき，ハードウエアあるいはインターフェースのあるプログラムなどの走行環境の違い，ＣＰＵ，メモリ，周辺装置などのハードウエア資源，あるいはメモリ常駐のデータやプログラムなどのソフトウエア資源への競合によって，期待した品質にならない場合が多い．そのため，通常負荷あるいは高負荷状態での長時間にわたる安定化試験により，サービス開始後の負荷に対して，応答時間そしてＣＰＵ，メモリ，周辺装置などの資源の十分性を事前



に確認しておく必要がある．また，この試験を通して，サービス開始後の平均故障時間間隔（ＭＴＢＦ：Mean Time Between Failures）およびＭＴＢＤを推定し，必要に応じてさらに品質向上策を打つ判断も可能となる．その場合，実際の運用と同じ環境，同じデータでの試験が最良であるが，そのような試験は，一般には膨大な稼働を必要とする．このため，試験対象システムの端末台数や試験データを縮小した試験で済ます場合が多い．

また，大規模システムは，一旦開発されるとシステム更改を重ねながら使用される．その更改の大半は，従来機能への新規機能の追加である．この場合，従来の大量の応用プログラム（ＡＰ）とデータベース（ＤＢ）の互換性を保証する必要がある．しかし，その保証にも大量の試験データと試験稼働が必要である．そして，性能面ではサービス開始直後から従来と同等の負荷がかかるため，総合試験では従来の最大負荷を想定した試験も必要である．

このように大規模システムの総合試験では，試験時間の短縮，試験稼働の削減が重要な課題である．

表６．１は，実際に運用開始後に検出されたフォールト８３件を，検出遅延要因という観点から分析したものである．品質を最終的に保証する総合試験を進める場合，過去のシステムで検出されたフォールト内容も大いに参考になる．同種のフォールトが残存している可能性が高いからである．信頼性を保証するためには，次の試験データや試験環境が必要であることが，このフォールトの分析から得られる．

（１）多種多様な試験データ

（２）いろいろなタイミングを生成できる試験環境

（３）実運用に則した試験環境

多種多様な試験データは，表６．１の項番１と項番５のフォールトに対処するものである．いろいろなタイミングを生成できる試験環境は項番２のフォールト，実運用に則した試験環境は項番３のフォールトに対処するためのものである．

これらの試験データおよび試験環境の準備も総合試験の課題である．

表６．１　運用中に検出されたフォールト

| 項番 | 検出遅延原因 | 件数 | 比率 | フォールト内容 |
|---|---|---|---|---|
| 1 | 試験データのバリエーション不足 | 40件 | 48% | (1) 入力データのバリエーション<br>(2) コマンドのパラメータのバリエーション<br>(3) ファイル内容のバリエーション<br>(4) 利用者契約情報のバリエーション |
| 2 | 試験でタイミングをつかむのが難しい | 21件 | 25% | (1) ファイルへのロックのタイミング<br>(2) タスク間イベントの交換のタイミング<br>(3) 非同期割り込みのタイミング<br>(4) タスクのイベント処理とセンタコマンドのタイミング |
| 3 | 試験環境がサービス時環境と異なる | 12件 | 15% | (1) 試験負荷が低かった<br>(2) 試験時のファイル量が少ない<br>(3) 商用と違うモードでの試験<br>(4) センタにディペンドするハード構成に対する試験不足 |
| 4 | テスト結果のチェックが不十分 | 7件 | 8% | (1) プログラムワーク用の内部テーブルの状態チェック不足<br>(2) 端末へのメッセージ情報のチェック不足 |
| 5 | テスト用に特異データを作成しないと検出できない | 3件 | 4% | (1) 大量データ処理時の配慮不足<br>(2) データの区切りが特定の値になった時の配慮漏れ |



## ６．２　総合試験の役割と試験システムの条件

### ６．２．１　総合試験の役割

総合試験までに実施される単体試験および統合試験では，プログラムのキロライン当たりの残存フォールト数，あるいは製造時に混入したフォールトの何パーセントを摘出しておくかという目標管理の下で，開発機能全体を網羅的に確認する試験が主である．これに対して，総合試験では，システム全体の最終的な品質保証のために，以下の確認を行う．

（１）要求機能を満足し，システムが長時間安定して動作することの確認

（２）ＣＰＵ使用時間，高負荷状態での応答時間，メモリ使用量，記憶装置の使用率など，性能面で問題のないことの確認

なお，システムの性能に関しては，実際の運用中に発生する最大負荷を確認しておくことが必要であり，さらに一歩進めて，システム資源それぞれの限界を見極めておくことも重要である．そのためには，実運用と同じ環境（ハードウェア構成，ソフトウェア構成，データベース保管状態）で試験をするのが必要条件となる．

また，サービス開始後の信頼性保証を確実にするために，それまでの試験では究明できないでいた障害原因の一つひとつの解明，および検出し難いフォールトの再現試験も実施する．

### ６．２．２　試験システムの条件

総合試験用の試験システムの機能は，（ａ）試験環境と試験データの作成機能，（ｂ）試験実行制御機能，および（ｃ）試験結果の確認機能，に区分できる．但し，ここでの試験環境の作成とは，試験対象システム上で走行するＡＰとＤＢを試験機上に設定することをいい，試験データはＡＰへの入力データをいう．

（１）試験環境と試験データの作成機能

表６．１の運用中に検出されたフォールトの分析結果が示すように，大規模システムのソフトウェアの信頼性向上には，いかに多種多様の試験データを用意できるかにかかっている．特に，システム更改では，従来機能

の互換性を保証するデグレード試験が不可欠である．このためのデータは，新たに作成するより，実運用で使用されるデータを利用する方がより多様なデータを得ることができる．運用中のデータを利用するためには，運用システムのＡＰ，ＤＢ，そして，システムと端末との交信データを収集する機能が必要である．

（２）試験実行制御機能

長時間の安定動作確認試験を実施するには，試験データを長時間，高負荷で試験対象システムに供給できる機能が必要となる．この場合，供給する試験データ量を，簡便に制御できる機能も合わせて必要となる．以下，単位時間当たりのシステムへのデータ転送量と同時接続端末数との積をトラヒックと呼ぶ．

（３）試験結果の確認機能

総合試験では，試験対象システムへ入力したデータに対する応答データの合否を確認する必要がある．大量の入力データを用いる場合，この確認作業を人手で実施するのは，作業量，確認時間共に大きくなり，現実的ではない．そのため，試験システムでは，試験結果を自動照合する機能が必要となる．当該機能は，実運用で使用された応答データを正解値として，試験時の応答データを，それと照合させることにより実現可能である．

## ６．２．３　試験効率化モデル

前述のように，総合試験では試験時間の短縮が重要な課題である．実運用中のデータを試験データとして用いた場合の試験時間の短縮は，センタ・端末間のデータ入力から処理結果受取りまでの時間，これをインタラクション時間と呼ぶ，の短縮とトラヒック制御による運転時間の短縮によって可能となる．

（Ａ）インタラクション時間の短縮

センタ・端末間のインタラクション時間χは，図６．１のモデルで表すことができる．すなわち，

$$\chi = \alpha + \beta + \gamma + \delta \qquad (6.1)$$

ここで，α：端末利用者の思考時間



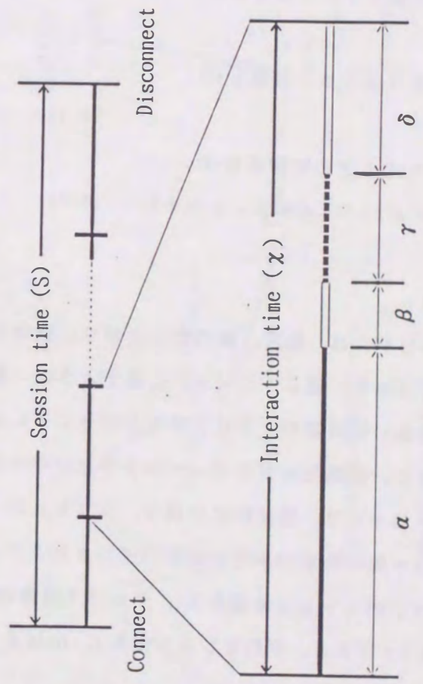



図６．１　インタラクションモデル

$\beta$：端末からセンタへのデータ転送時間

$\gamma$：センタの処理時間

$\delta$：センタから端末へのデータ転送時間

一方，試験時のインタラクション時間$\chi_T$は，試験システムでは制御できない$\gamma$，試験システムのハードウエア構成によって決定される$\beta$および$\delta$，試験システムのトラヒック制御機能で制御可能な$\alpha$，そして，試験システムの処理時間$\varepsilon$からなる．ここで，センタ・端末間を高速の通信回線で接続すれば，$\beta$と$\delta$は，転送データ量が非常に大きい場合を除けば無視できる．さらに，$\alpha$は試験システムの制御により無視できるほど小さくできる．

よって，試験時のインタラクション時間$\chi_T$は，

$$\chi_T = \gamma + \varepsilon \qquad (6.2)$$

まで短縮可能であり，そのときの短縮率$\Theta_T$は，

$$\Theta_T = \chi_T / \chi = (\gamma + \varepsilon) / (\alpha + \beta + \gamma + \delta) \qquad (6.3)$$

となる．

（B）運転時間の短縮

運用中の一日のトラヒックは，通常，朝の立ち上がり，昼休み，および夕方から夜間にかけては低く，図６．２（ａ）に示すように一様ではない．これらのトラヒックの低い時間帯のトラヒックを上げることにより，試験時間の短縮が可能となる．試験対象システムへのトラヒックを制御することにより，試験のトラヒックは，理想的には図６．２（ｂ）のようにできる．一日のセション（一連の処理の単位で複数のインタラクションからなる）数をｎ，セションｊのセション時間を$S_j$，サービス開始時刻を$t_s$，サービス終了時刻を$t_\varepsilon$とすると，平均セション時間$S_{av}$は次式となる．

$$S_{av} = \sum_{j=1}^{n} S_j / n \qquad (6.4)$$

システム運用時の平均同時接続数$C_{av}$は，

$$C_{av} = \sum_{j=1}^{n} S_j / (t_\varepsilon - t_s) \qquad (6.5)$$

一日の運転時間Ｔは，



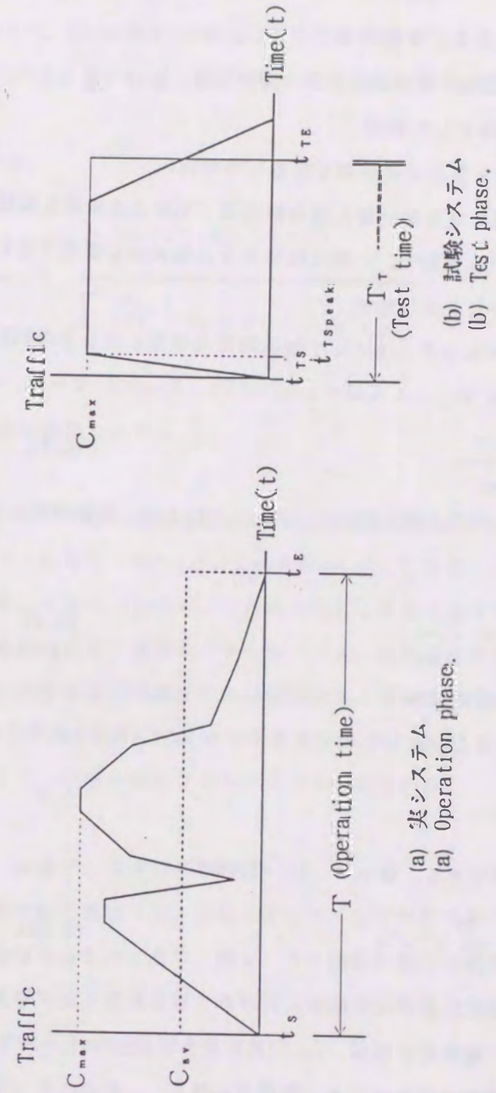



図６．２　システムトラヒック特性

$$T = t_E - t_S = \frac{S_{av} \times n}{C_{av}} \qquad (6.6)$$

と表すことができる．また，

$t_{TS}$：試験を開始した時刻，

$t_{TE}$：試験を終了した時刻，

$t_{TSpeak}$：試験トラヒックが高く安定した時刻，

$C_{max}$：過負荷にならない最大同時接続数（平均応答時間を保証するために用いる定数で，一般にはシステム生成時に設定する），

$S_{max}$：最長のセション時間，

とすれば，トラヒックを高くして運転時間を短縮したときの試験時間$T_T$（$= t_{TE} - t_{TS}$）は，（$t_{TSpeak} - t_{TS}$）$\ll T_T$，$S_{max} \ll T_T$ならば，近似的に，

$$T_T \fallingdotseq \frac{S_{av} \times n}{C_{max}} \qquad (6.7)$$

となる．よって，同時接続数を上げることにより，試験時間は次式のように短縮できる．

$$\Theta_C = \frac{T_T}{T} = \frac{C_{av}}{C_{max}} \qquad (6.8)$$

（C）試験時間短縮率

いま，j番目のセションのインタラクション回数を$m_j$とすれば，

$$S_j = \sum_{i=1}^{m_j} \chi_{ij} \qquad (6.9)$$

と表すことができる．従って，(6.4)式は，

$$S_{av} = \sum_{j=1}^{n} \sum_{i=1}^{m_j} \chi_{ij} / n \qquad (6.10)$$

となる．同様に，$\chi_{Tij}$を試験時におけるj番目のセションのi番目のインタラクション時間とすれば，(6.2)式の最も短縮したインタラクション時間を用いた試験時の平均セション時間$S_{Tav}$は，

$$S_{Tav} = \sum_{j=1}^{n} \sum_{i=1}^{m_j} \chi_{Tij} / n \qquad (6.11)$$

となるので，(6.7)式は，



$$T_T \fallingdotseq \frac{S_{Tav} \times n}{C_{max}} \qquad (6.12)$$

となる．従って，(6.8)式の$T_T$を(6.12)式で置き換えて，(6.1),(6.2),(6.10),(6.11)式を用いれば，試験時間の理想的な短縮率$\Theta$は次式となる．

$$\Theta = \Theta_C \times \Theta_T = \frac{C_{av}}{C_{max}} \times \frac{\sum_{j=1}^{n} \sum_{i=1}^{m_j} (\gamma_{ij} + \varepsilon_{ij})}{\sum_{j=1}^{n} \sum_{i=1}^{m_j} (\alpha_{ij} + \beta_{ij} + \gamma_{ij} + \delta_{ij})} \qquad (6.13)$$

## 6．3　総合試験用システム

筆者らは，オンライン情報処理システムの性能試験を目的にした多端末シミュレータ（ＭＴＳ：Multi-Terminal Simulator）など，一連の試験システムを開発してきた[1]～[4]．その集大成として図6．3に示す総合試験システムを開発した．当該システムは，（a）試験データ事前収集機構，（b）機能試験や安定化試験などの信頼性向上と保証に使うＲＡＳＰ（Reliability Assurance System using Practical data），（c）性能試験に使うＭＴＳ，の三つのサブシステムから構成される．

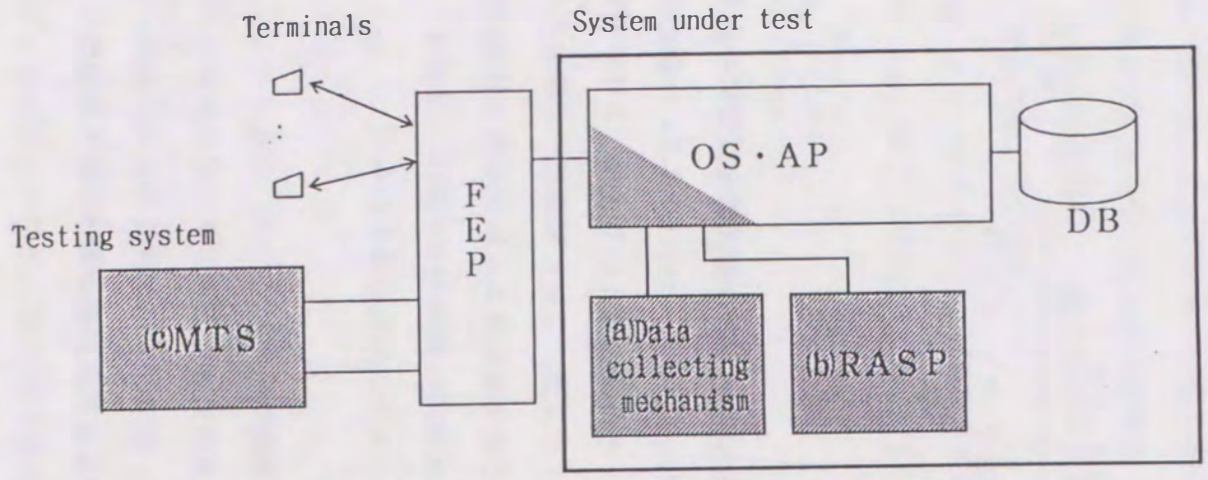
### 6．3．1　試験データ事前収集機構

試験データ事前収集機構は，実際にオンラインサービスを行っているシステム内で動作するものであり，図6．4の網掛け部分に当たる．当該機構は，入出力データの収集そしてＡＰおよびＤＢの収集の２つからなる．

（1）入出力データの収集

（A）入力データの収集

入力データの収集は通信制御プログラムを改造して実現している．システム内の図6．4（a）通信制御プログラムが，端末からの入力データをＴＳＰ／ＲＴＰに渡すとき，同時に図6．4（b）データ収集プログラ

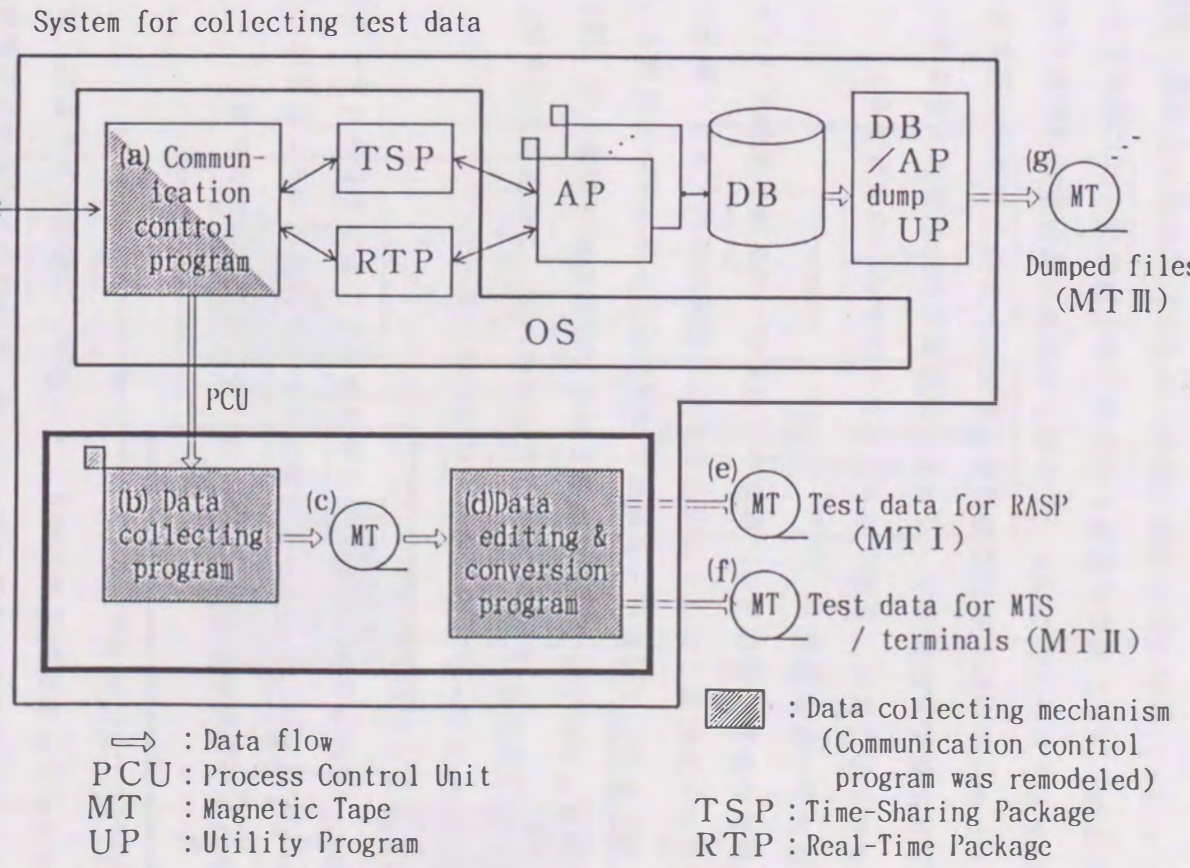



MTS : Multi-Terminal Simulator
FEP : Front End Processor
OS : Operating System
RASP : Reliability Assurance System using Practical data

AP : Application Program
DB : DataBase
: Testing system
[ A part of OS was remodeled ]

図6．3　総合試験システムの構成






⇨ : Data flow
PCU : Process Control Unit
MT : Magnetic Tape
UP : Utility Program

TSP : Time-Sharing Package
RTP : Real-Time Package

図6．4　試験データ収集機構の概要

ムにも渡す．渡すデータはＰＣＵ（Process Control Unit）と呼び，前置処理装置（ＦＥＰ：Front End Processor）と通信制御プログラムとの交信単位である．

１インタラクションは，通常，複数のＰＣＵからなる．データ収集プログラムは，収集したＰＣＵを端末別およびセション別に，時系列に整理して図６．４（ｃ）ＭＴに出力する．図６．４（ｄ）試験データ編集／変換プログラムは，（ｃ）ＭＴに収集されたデータを試験データに編集する．ひとつはＲＡＳＰの入力形式に，もうひとつは通信制御情報の削除や分割された入力データの結合などの処理をして端末からの入力形式に編集する．このデータはＭＴＳの入力データあるいは実端末から直接入力するデータとして使用する．なお，当該プログラムはオフラインで走行する．

（Ｂ）出力データの収集

試験結果の合否の確認用に，端末への出力データも通信制御プログラムを通してデータ収集プログラムで収集する．当該出力データは，図６．４（ｃ）ＭＴを介して試験データ編集プログラムで，入力データと対にして，ＲＡＳＰおよびＭＴＳでの試験結果と照合できる形式に変換して，図６．４（ｅ）ＭＴⅠおよび（ｆ）ＭＴⅡに出力される．この出力データは，試験実施時の出力データとの自動照合に用いられる．

（２）ＡＰおよびＤＢの収集

一般に，どのシステムにもＡＰとＤＢをＭＴに出力するユーティリティプログラムが備わっている．本試験システムの場合もＡＰとＤＢは，これらのプログラムを使用して，図６．４（ｇ）ＭＴⅢに出力する．

## ６．３．２　ＲＡＳＰ

ＲＡＳＰによる試験方式を図６．５に示す．本試験システムは，ＭＴＳのように外付けの試験設備を用いないで，実運用システムだけで構築される．従って，試験対象システムの処理能力を越えない限り，図６．１のインタラクションモデルでの思考時間，データ送信時間，およびデータ受信時間を最小にすると同時に，処理セション数を増大することが可能である．このため，最大限に時間短縮した長時間安定化試験やデグレード試験が可



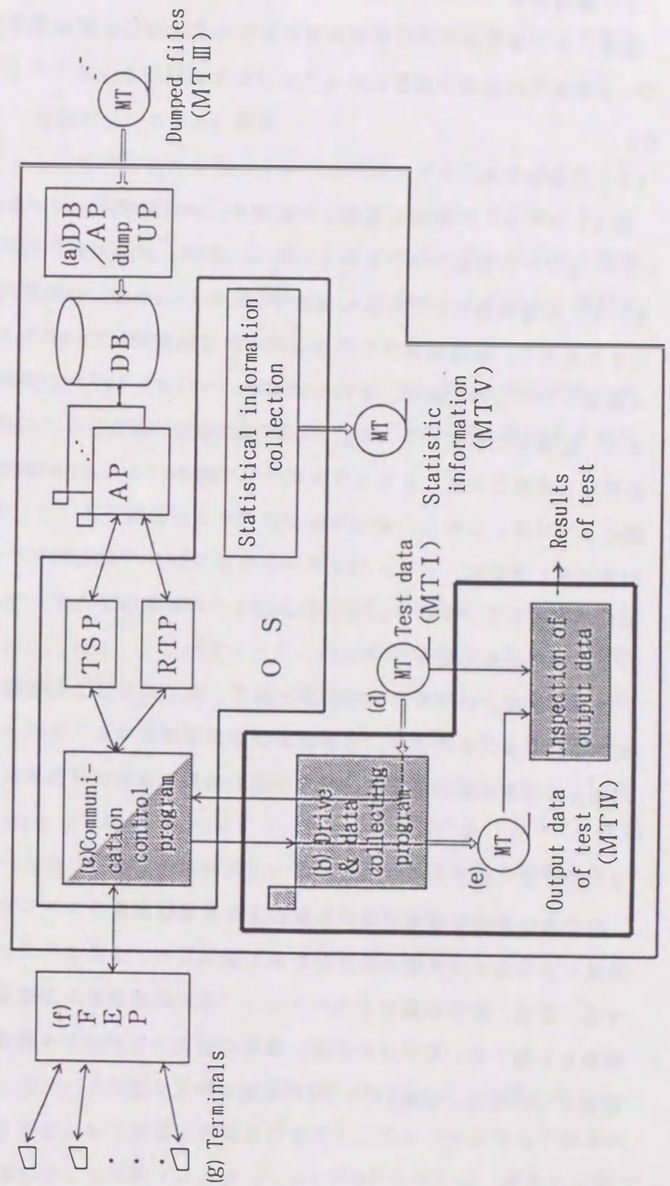



図６．５　ＲＡＳＰによる試験方式

能となる．ＲＡＳＰ試験は以下の手順で進める．

（１）事前処理

図６．５（ａ）ユーティリティプログラムによって，試験開始前に試験データ事前収集機構で収集したＡＰとＤＢを試験対象システム上に復元する．

（２）試験の実施

図６．４（ｅ）で事前に収集した試験データを，図６．５（ｂ）のドライバー＆データ収集プログラムによって，順次，試験対象システムの図６．５（ｃ）通信制御プログラムに送出する．ＡＰが処理した結果は端末に出力する直前で，通信制御プログラムが出力先を変更して，ドライバ＆データ収集プログラムに渡す．それらは図６．５（ｅ）ＭＴⅣに格納される．また，試験時に図６．５（ｆ）ＦＥＰ上の通信ソフトウエアの機能確認が必要なときのために，ＲＡＳＰと並行して図６．５（ｇ）の端末を動作可能にしている．これは，通信制御プログラムが論理パス（センタと端末との通信路）単位に，ドライバ＆データ収集プログラムに渡すデータか，端末に出力するデータかを区別することにより実現している．

（３）試験結果の確認

試験終了後，ＭＴⅣの処理結果と図６．５（ｄ）ＭＴＩの事前収集データを照合することにより，処理結果の合否を確認する．なお，端末を用いた場合の試験結果の確認は，端末の出力結果を人手で確認する．

### ６．３．３　ＭＴＳ

ＭＴＳは多数の端末の動作を擬似する性能試験用システムである．試験対象システムとは実際の複数の回線で接続され，多様なトラヒックを発生する．また，応答時間やスループット（単位時間当たりの処理量）の測定機能も具備する．ＭＴＳの構成，機能の詳細およびＭＴＳを用いた試験実施法については，文献[1]～[3]に記述されている．



## ６．４　適用例と評価

ここでは，ＲＡＳＰを用いた長時間安定化試験の実施結果について述べる．

### ６．４．１　試験対象システムの概要

対象システムは科学技術計算用タイムシェアリングシステム（ＴＳＳ）の更改版である．全国６箇所にセンタがあり，センタ間は専用線でネットワーク化されている．各センタには，図６．６に示すソフトウエアが搭載されている．ＡＰにはシステムが提供するライブラリプログラムと利用者が開発したユーザプログラムがある．

システムの更改では，既存のロードモジュール（計算機が実行可能な形式のプログラム）の実行保証が必要である．また，サービス開始直後から更改前と同等のトラヒックがある．

### ６．４．２　収集試験データ量

実際の運用中（サービス時間帯：午前８時～午後１０時，日曜日は運休）のシステムから，表６．２に示すセンタ・端末間の交信データを３０日間にわたって収集した．３０日のうち１８日は全データを，１２日は端末指定で収集した．全データを収集した日の平均ＰＣＵ数は，１９６，９７２（＝3,545,494/18）である．また，端末指定で収集した日の合計ＰＣＵ数は，６３４，４４８である．これは，３．２日分に相当する．よって，収集した試験データ量は，実質的には２１．２日分で，ほぼ２９７時間の量になる．

### ６．４．３　試験の効率化効果

（１）試験時間の短縮効果

収集した試験データのうち７日分（239,936ＰＣＵ）は，運用システムと比べて，ＤＫ装置，ＭＴ装置，回線制御装置の少ないハードウエア構成の試験機で使用した．残りの２３日分（3,941,006ＰＣＵ）は，運休の日曜日を利用して，実際の運用システムで使用した．前者の試験では，試験デー

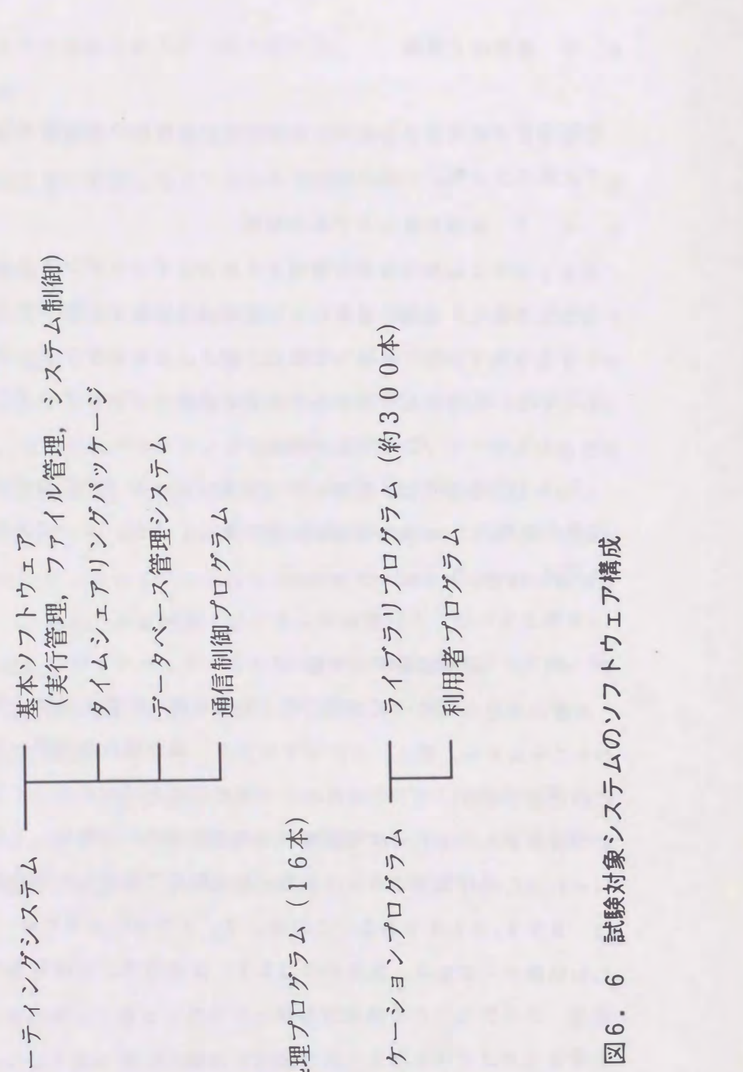



図6．6　試験対象システムのソフトウェア構成



表6．2　RASP試験データ

| 収集回数 | PCU数 | ｾｼｮﾝ数 | 記　事 |
|---|---|---|---|
| 1 | 14,234 | 86 | 注：10端末 |
| 2 | 22,267 | 164 | 注：26 〃 |
| 3 | 29,463 | 193 | 注：31 〃 |
| 4 | 33,731 | 217 | 注：31 〃 |
| 5 | 50,152 | 322 | 注：59 〃 |
| 6 | 41,252 | 230 | 注：85 〃 |
| 7 | 42,022 | 198 | 注：85 〃 |
| 8 | 56,966 | 346 | 注：85 〃 |
| 9 | 90,477 | 543 | 注：71 〃 |
| 10 | 82,880 | 570 | 注：86 〃 |
| 11 | 85,417 | 581 | 注：99 〃 |
| 12 | 86,587 | 539 | 注：48 〃 |
| 13 | 98,085 | 590 | |
| 14 | 105,175 | 704 | |
| 15 | 197,864 | 1,350 | |
| 16 | 216,406 | 1,281 | |
| 17 | 229,088 | 1,202 | |
| 18 | 258,510 | 1,438 | |
| 19 | 166,912 | 1,037 | |
| 20 | 246,013 | 1,516 | |
| 21 | 230,779 | 1,510 | |
| 22 | 270,374 | 1,528 | |
| 23 | 177,670 | 1,026 | |
| 24 | 155,685 | 971 | |
| 25 | 139,870 | 937 | |
| 26 | 253,200 | 1,602 | |
| 27 | 150,139 | 1,057 | |
| 28 | 170,031 | 1,073 | |
| 29 | 223,944 | 1,487 | |
| 30 | 255,749 | 1,423 | |
| 累計 | 4,180,942 | 25,721 | 163PCU/セション |

注〕：端末指定により一部データのみ収集

タを収集した運用システムと試験機システムとのシステム構成の違いなどにより，約１２．６％の試験データが無効になった．後者の試験では，試験時間の制約があり，途中で試験を打ち切るなどにより，２６．０％の試験データが未使用となった．その結果，収集したＰＣＵ数4,180,942に対して，使用したＰＣＵ数は3,126,852で，収集データの７４．８％に当たり，19,200（＝3,126,852／163）セションに相当する．また，実施した試験時間は合計８２時間３０分であった．

よって，試験時間は，１／２．７（＝82.5/(297×0.748)）に短縮できたことになる．

一方，試験データ収集時と試験実施時の平均同時接続数の比，すなわち，(6.8)式の$\Theta_c$は，およそ０．５である．また，インタラクションの平均的特性は，

・実際の平均インタラクション時間$\chi \fallingdotseq 40$秒

・平均センタ処理時間$\gamma \fallingdotseq 6$秒

・ＲＡＳＰの平均処理時間$\varepsilon \fallingdotseq 1$秒

・ＲＡＳＰのトラヒック制御時間$\alpha = 0$秒，$\beta \fallingdotseq 0$秒，$\delta \fallingdotseq 0$秒

であるので，$\Theta_T$は，およそ１／６（＝(6+1)/40）である．従って，(6.13)式より理想的な試験時間短縮率$\Theta$は１／１２となる．

ところで，別途実施した性能試験から，試験対象システムは約60,000ＰＣＵ／時間（セション数に換算すると約３６８セション／時間）の処理能力を有するデータが得られた．この処理能力からすると，試験は５２．２（＝19,200／368）時間で完了できることになり，これを短縮率で表すと，１／４．３（＝52.2/(297×0.748)）になる．

今回の試験の短縮率が，１／２．７となった要因として，次の２点が挙げられる．

（Ａ）(6.1)式の$\alpha$を零としたため，試験対象システムに過大な負荷がかかり，$\gamma$と$\varepsilon$が見込み値より大きくなった．

（Ｂ）実際の試験では図６．２（ｂ）の垂直線のように終了せず，斜線のように徐々にセションが終了するため試験時間が延びた．



従って，今後の改善点として，試験システムでシステムの負荷を監視し，その値に応じて$\alpha$を自動制御する機能を具備することが挙げられる．これにより過負荷による処理能力の低下を防止でき，短縮率はさらに向上すると考えられる．

（２）試験稼働等の削減効果

試験稼働は，大きく次の４項目に分類できる．

（Ａ）データ収集・作成稼働

（Ｂ）試験環境設定稼働

（Ｃ）試験実施稼働

（Ｄ）試験後の確認および後処理稼働

ＲＡＳＰ導入前の（Ｃ）試験実施稼働を１として，これまでの経験から得られた，ＲＡＳＰ導入前後の試験稼働の比較を表６．３と図６．７に示す．

まず，（Ａ）について，ＲＡＳＰの導入前の場合，システム更改時のデグレード試験は，既存システムの総合試験データを使用することで済ませていた．ＲＡＳＰ導入後のように新たなデータの追加は行われていない．従って，導入前は従来データの利用なので稼働は零，導入後は新規データを作成する作業が増えることになる．（Ｂ）はＡＰおよびＤＢの設定作業が主であるため，同等と見ることができる．（Ｃ）および（Ｄ）は自動化の効果が顕著な部分である．稼働の削減効果を論じる場合，（Ａ）は比較の意味がないため対象外とすると，試験稼働は従来の１／５．８（≒0.26/1.50）となり，効果は大であるといえる．

### ６．４．４　試験の信頼性向上効果

ＲＡＳＰ適用試験で検出したフォールト１５件を，プログラムの種類と検出遅延要因で分類したものを表６．４に示す．１５件のうち１４件（表６．４のＡ，Ｂ，Ｃ項のフォールト）は，従来の試験環境とデータでは検出し難いフォールトであり，実際の運用で投入されたデータを高負荷状態で用いて試験した効果である．

また，ＲＡＳＰを用いて試験したシステムと，このシステム更改前の

表６．３　試験稼働の比較

| 試験稼働 | 従来試験 | ＲＡＳＰ適用 |
|---|---|---|
| （A）データ収集・作成 | 0 | 0.07 |
| （B）試験環境設定 | 0.12 | 0.12 |
| （C）試験実施 | 1 | 0.07 |
| （D）試験後の確認・後処理 | 0.38 | 0.07 |
| 合　　計 | 1.50 | 0.33 |

（注）従来試験実施稼働を１としたときの相対値である．

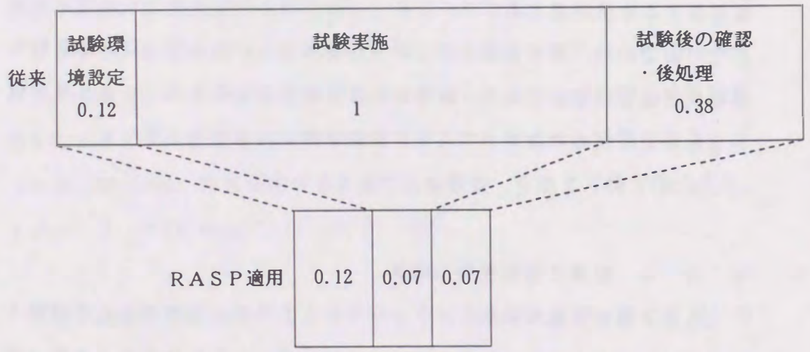



図６．７　試験稼働の比較



表６．４　ＲＡＳＰ試験での検出フォールト

| 検出遅延原因 | ＯＳ | ＬＰ | ＡＰ | 合計 |
|---|---|---|---|---|
| （A）試験データのバリエーション不足 | 2 | 1 | 4 | 7 |
| （B）タイミングをつかむのが難しい | 5 | 0 | 0 | 5 |
| （C）試験環境が実運用環境と異なる | 2 | 0 | 0 | 2 |
| （D）テスト結果のチェックが不十分 | 1 | 0 | 0 | 1 |
| （E）特異データを作成しないと検出不可 | 0 | 0 | 0 | 0 |
| 合　　計 | １０ | 1 | 4 | １５ |

表６．５　ＲＡＳＰ導入前後のＭＴＢＤの比較

| サービス開始後の期間 | ＭＴＢＤ | |
|---|---|---|
| | 導入前 | 導入後 |
| 直後一ヶ月目 | 2.1 | 5.9 |
| 直後二ヶ月目 | 1.8 | 2.8 |
| 直後三ヶ月目 | 2.3 | 5.5 |
| 平　均 | 2.1 | 4.7 |

（注）過去のデータから最低値を１としたときの相対値

ＲＡＳＰを用いないシステムとのサービス開始後３か月間のＭＴＢＤの比較を表６．５に示す．ＭＴＢＤはＲＡＳＰ適用の方が約２倍ほど長くなっている．効果があったともいえるが，これはＲＡＳＰによる試験実施までに確保した信頼性にも依存するため，一概にＲＡＳＰ適用効果ということはできない．

## ６．５　フォールト検出難易度でみた信頼性

表６．１のフォールトを，フォールト検出難易度で分類すると，項番４はクラス１，項番１，３，５はクラス２，項番２はクラス３に分類される．従って，クラス１，２，３の相対検出率は，それぞれ 0.08，0.67，0.25 となる．表６．１のソフトウェアはシステムプログラムに属する．そこで，これらの値を表４．４に示すシステムプログラムの値と比較すると，総合試験終了時点（$J_2$）の最新相対検出率（0.1，0.7，0.2）と同種の安定運用中のソフトウェアの最新相対検出率（0，0.7，0.3）との間に位置する．表６．１のフォールトは，運用開始後に検出されたフォールトであり，ＦＤＭの根拠とするフォールト特性と合致する．また，表６．１のデータと表４．４のデータとの関係は，同種のソフトウェアの相対検出率は似通った特性を示すこと，そして，開発済みのソフトウェアのフォールトデータは，同種の開発中のソフトウェアの評価に利用できることを意味する．

表６．３のフォールトを表６．１と同様に，フォールト検出難易度で分類すると，クラス１，２，３の相対検出率は，0.07，0.60，0.33となる．従って，ＦＤＭによる評価では，表６．３のソフトウェアは，表６．１のソフトウェアよりも少し信頼性が高いことになる．この評価結果は，表６．５に示すＭＴＢＤのデータによって裏付けられる．逆に，ＦＤＭを用いれば，表６．５の比較結果が推定できるともいえる．

結果的に，表６．１および表６．３のフォールトデータは，ＦＤＭの評価精度の確かさを裏付ける．



# 第7章　結論

## ７．１　連結指数形ソフトウェア信頼度成長モデル

### ７．１．１　信頼度成長曲線形成の物理的解釈

フォールト検出率（単位時間当たりのフォールト検出数）を用いて実際の信頼度成長曲線の特性を分析し，統合試験，総合試験，運用それぞれの工程で，指数形あるいはＳ字形の信頼度成長曲線が形成されていることを示した．さらに，これら全体の工程を通した巨視的な成長曲線の特性はＳ字形であることも示した．そして，各試験工程の進め方をモデル化し，試験工程の進め方と形成される成長曲線の特性との関係を明らかにし，つぎの新しい物理的解釈を述べた．すなわち，

（１）信頼度成長曲線の基本特性は指数形であるが，メインルート構成モジュールが先行して統合されるという統合試験の進め方から，Ｓ字形が形成される．

（２）全試験項目を確認するという試験目的のために，試験工程終盤になるとフォールト検出率が本来より低くなる．

そして，これまでのＳＲＧＭ論では殆ど触れられていない成長曲線の試験工程境目での段丘形状の特性について，これが上記（２）に起因しており，大規模ソフトウェアの開発で形成されることを述べ，事例を示した．

### ７．１．２　連結指数形ＳＲＧＭと近似式の導出

上記の物理的解釈から，大規模ソフトウェアの信頼度成長曲線は，小さな指数形成長曲線と，それに続く大きな指数形成長曲線とが連結して出来ることを述べ，連結指数形ＳＲＧＭを導出した．そして，連結指数形ＳＲＧＭが形成する成長曲線は，実際にはＳ字形として観測されること，さらにソフトウェアの規模が小さくなると，指数形の特性曲線になることを述べた．

当該モデルを開発現場で使用するには，メインルート構成モジュールの統合と残りの大多数のモジュールの統合との分界点を明確にして進める必要がある．しかし，これは開発上の大きな制約となる．そこで，指数形ＳＲＧＭを近似式として用いることを提示した．連結指数形ＳＲＧＭと指数形ＳＲＧＭとの推定値の差は，測定数を１０以上にすると，ほとんど無視できる程度になることを示し，指数形ＳＲＧＭが近似式として使用できることを述べた．

### ７．１．３　連結指数形ＳＲＧＭの適用法と推定精度

試験工程の終盤と開始時期はフォールト検出率が低くなるという物理的解釈に基づき，それに該当する測定データをノイズデータとみて，適用データからふるい落とすことにより推定精度を向上させるという適用方法を述べ，データふるい落としアルゴリズムを述べた．

そして，実際の大規模システムの開発データを用いて，代表的な３つのＳＲＧＭと残存フォールト数の推定精度を比較し，連結指数形ＳＲＧＭは，試験工程半ばの統合試験終了時点での推定精度が２～１０倍良いことを示した．

### ７．１．４　今後の課題

連結指数形ＳＲＧＭは，次式で表わされる．

$$M(t)=a_1[1-\exp(-bt)]+a_2[1-\exp(-by)], \qquad (2.5)$$

$$a_2>>a_1>0,\quad b>0,$$

$$y=\begin{cases}0 & t<t_0,\\ t-t_0, & t\geqq t_0.\end{cases}$$

ここでは，フォールト出現率bは一定と仮定している．しかし，実際のソフトウェア開発では，試験の進捗と共に，検出困難なフォールトの比率が多くなってくる．すなわち，統合試験より総合試験の方がフォールト検出率（単位時間当たりのフォールト検出数）が低くなるのは，指数形ＳＲＧＭの物理的解釈である残存フォールト数が少なくなったからだけではなく，出現率の低いフォールトの比率が多くなった要因も影響していると考えられる．従って，試験の進捗と共にフォールト出現率が低減していく連結指数形ＳＲＧＭを導出すれば，より高い精度で残存フォールト数を推定できると考えられる．すなわち，上記モデルの導出と適用法の開発が，今後の課題として挙げられる．



## ７．２　フォールト検出難易度モデル（ＦＤＭ）

### ７．２．１　フォールト検出難易度の分類基準

タイミングを現出させる実行多重度および実行環境や特異な処理データの実行条件という２つの分類軸を用いて，フォールト検出難易度を３クラスに分類する分類基準と手順を述べ，各クラスのフォールト例を示した．また，各検出難易度クラスのフォールト発生メカニズムをペトリネットを用いてモデル化し，フォールト検出難易度の違いを明らかにした．そして，実際に運用されたソフトウェアの運用２年間に検出されたフォールトの分類事例を示した．

### ７．２．２　ＦＤＭの導出と推定精度

検出フォールト数の増大，換言すれば残存フォールト数の減少と共に，検出難易度の低いフォールトの検出割合は下がり，逆に高いフォールトの検出割合が大きくなっていくという物理的解釈を述べた．この物理的解釈に基づき，最新ｍ個のフォールトにおける検出難易度クラス別比率，すなわち最新相対検出率から残存フォールト数を推定するＦＤＭを導出した．但し，最新相対検出率を求めるときの母数とするフォールト件数は，１０件以上を必要とすることも述べた．

そして，実際に開発されたソフトウェアのフォールトデータを用いて，ＦＤＭおよび代表的な３つのＳＲＧＭによる残存フォールト数推定値を求めた．それらを比較して，統合試験終了時点でのＦＤＭの推定精度は，他のＳＲＧＭより７～９倍高いことを示した．

### 7．2．3　残存フォールト数の実践的推定法

実用ソフトウェアの開発において，残存フォールト数を実数より過少評価すると，進捗上の大きな問題を引き起こす場合がある．対策は後手後手になり，試験の収束までに予期せぬ時間と費用がかかる．

少し多めの稼働を要するにしても，むしろ，多少過大気味の残存フォールト数の評価の方がリスクは小さい．従って，信頼性評価モデルを適用する場合，一時的なフォールト特性で残存フォールト数を過少評価していないかの点検が重要となる．

ＦＤＭの物理的解釈に基づけば，分析対象のフォールト件数を増加させると，それだけ時間的に古いフォールトが多くなるため，残存フォールト数は多めに推定される．常にこれが成り立つとは限らないが，この特性を利用すれば，リスクを小さくする推定値が得られる．以上の考えに基づき，分析対象のフォールト件数を変えて，２つの残存フォールト数の推定値を算出し，ヒューリスティックな考察でリスクの小さな残存フォールト数を決定するという実践的適用法を述べた．

### 7．2．4　ＦＤＭによる信頼性成長の図形表記

開発ソフトウェアと同種のソフトウェアの最終相対検出率および固有相対検出率のデータがあれば，検出フォールト数の割合Ｒを前もって全て計算することができる．そして，それらは正三角形上に割り付けることができる．ＦＤＭチャートと呼ぶその正三角形を提示した．この正三角形上に開発中のソフトウェアの最新相対検出率を表示していけば，信頼性の成長状況を図形として表示できる．

統合試験，総合試験では，それぞれフォールト検出目標値を定めて試験を進める．ＦＤＭチャート上に，その目標値および同種のソフトウェアの最終相対検出率をあらかじめ表示しておけば，目標への接近状況および目標到達状況を図形として把握可能となる．

ＦＤＭチャートは，開発現場での簡易な目標管理の道具として利用できることを述べた．



### 7．2．5　今後の課題

大規模ソフトウェアの開発において，統合試験終了時点での残存フォールト数は，総合試験における製造側のバックアップ要員数や体制，あるいは品質強化試験の必要性を決定する重要な指標である．この時点の残存フォールト数を精度よく推定できるＦＤＭは，開発管理の有用な道具となる可能性をもつ．

ＦＤＭ適用上の問題は，各種のソフトウェアの固有相対検出率および最終相対検出率のデータ蓄積がないことである．フォールトデータは，一般にソフトウェア故障処理票と呼ばれる帳票に記録される．ソフトウェア開発中は，これらの帳票はきちんと管理されるが，開発が終了し，しばらく時間が経過すると処分されてしまう．帳票の処分前に，上記データの収集，蓄積を実行できるかが課題である．

また，ＦＤＭの適用事例は，まだまだ少なく，現時点は，実用可能性を示した段階といえる．ＦＤＭの実用性を立証するには，幅広い追試が必要であり，そのためにも，上記データの収集，蓄積が重要である．そして，これらのデータ蓄積により，残存フォールト数の推定精度もまた，上がることが期待できる．

## 7．3　大規模オンラインシステムの高信頼化試験法

### 7．3．1　総合試験システムの機能と構成

大規模システムの総合試験では，実際の運用環境と同じ環境での試験が重要であり，そこで，長時間運転の確認および応答時間や処理能力の最終的な確認を行う必要がある．しかし，それらの試験には，大量の多種多様な試験データおよび膨大な試験実施稼働が必要であり，それらの効率化が課題となっている．

一般に，大規模システムは，一度開発されると更改を繰り返しながら長期間使用される．そのようなシステムの開発においては，実際に運用されているシステムのセンタ・端末間の交信データを試験データとして，そして，そこに搭載されているＡＰおよびＤＢを試験環境として使用する試験

方式が有効である．その試験方式を具体化した試験システムＲＡＳＰの機能と構成および処理方式について述べた．

### ７．３．２　試験効率化モデル

大規模システムの総合試験の効率は試験時間の短縮率で評価できる．試験時間はセンタ・端末間のインタラクション時間とトラヒックに依存するため，これらの時間の短縮が重要となる．

実運用におけるインタラクション時間とトラヒック特性を分析し，それぞれの時間短縮方法と短縮率を導出した．そして，試験時間の短縮率は，インタラクション時間の短縮率とトラヒック増大による運転時間の短縮率の積で表わされることを示した．

### ７．３．３　試験の効率と効果

ＲＡＳＰを用いた総合試験の評価として，試験時間の短縮率，試験稼働の削減率，そして高信頼化効果について述べた．すなわち，試験時間は１／２．７に短縮でき，試験稼働は従来と比較して１／５．８に削減できたことを示した．また，実際の試験時間が理想的な短縮率まで短縮できなかったことは，システムの負荷制御が十分でなかったことが要因として挙げられるという考察を述べた．

試験の高信頼化効果については，検出したフォールト１５件中の１４件が，従来の試験では検出できないフォールトであること，さらに，サービス開始直後の三ヶ月間のＭＴＢＤを従来と比較し，一概にＲＡＳＰの効果ということはできないが，約２倍の改善が見られたというデータを示した．

### ７．３．４　ＦＤＭの評価精度

表６．３および表６．１のフォールトをフォールト検出難易度で分類すると，クラス１，２，３の相対検出率は，それぞれ（0.08，0.67，0.25），（0.07，0.60，0.33）となる．従って，ＦＤＭによる評価では，表６．３のソフトウェアは，表６．１のソフトウェアよりも少し信頼性が高いことになる．この評価結果は，表６．５に示すＭＴＢＤのデータによって裏付けられる．



また，表６．１のソフトウェアはシステムプログラムに属するため，その相対検出率を表４．４に示すシステムプログラムの値と比較すると，総合試験終了時点（$J_2$）の最新相対検出率（0.1，0.7，0.2）と同種の安定運用中のソフトウェアの最新相対検出率（0，0.7，0.3）との間に位置する．表６．１のフォールトは，運用開始後に検出されたフォールトであり，これらの値の関係は，ＦＤＭが根拠とするフォールト特性と一致する．

表６．１および表６．３のデータは，また，同種のソフトウェアは類似のフォールト特性を有することを示す．これより，同種のソフトウェアの固有相対検出率および最終相対検出率を使用すれば，精度の高い残存フォールト数の推定が得られることが伺える．結果的に，表６．１および表６．３のフォールトデータは，ＦＤＭの評価精度の確かさを裏付ける．

### ７．３．５　今後の課題

大規模という言葉が意味する規模の大きさは，技術の進歩と共に変化している．１９７０年代は数百キロラインであり，１９８０年代は，これが１メガラインになり，１９９０年代は数メガライン以上になっている．最近，クライアント／サーバ（Ｃ／Ｓ）システムやネットワーク分散システムが開発されてきているが，大規模システムは，現状，センタ中心の構成が大半である．従って，ここで述べたＲＡＳＰ試験方式は，今後も，適用可能であり，しかも規模が大きくなるほど効果を発揮すると考えられる．

しかしながら，ネットワーク技術の進歩と共に，Ｃ／Ｓシステム，ネットワーク分散システムが増加してくるのは明らかである．これらネットワークを使用したシステムの品質評価，高信頼化を，効率よく実施できる総合試験方式と試験システムの開発が，今後の課題として挙げられる．

# 謝　辞

本論文をまとめるに当たり，懇切丁寧なご指導とご高配を賜った九州大学大学院システム情報科学研究科 牛島和夫教授に厚くお礼申し上げます．また，本論文について有益なご助言を賜った九州大学大学院システム情報科学研究科 牧之内顕文教授並びに程京徳教授に深く感謝申し上げます．

本研究は，筆者が日本電信電話株式会社横須賀電気通信研究所並びにソフトウェア研究所在職中に行ったものであり，この間，多数の先輩，同僚の方々からご指導ご鞭撻を頂きました．特に，ＮＴＴソフトウェア研究所 竹中市郎主幹研究員（現在久留米工業大学教授）にはフォールト検出難易度モデルの共同研究と共に研究全般に渡る多大なご指導を頂きました．ＮＴＴソフトウェア研究所 伊土誠一所長並びにＮＴＴ情報通信研究所 杉村康主任研究員には大規模オンラインシステムの高信頼化試験法の共同研究と共に暖かいご支援を頂きました．ＮＴＴソフトウェア研究所 古山恒夫主幹研究員（現在東海大学教授）にはソフトウェア信頼度成長統合モデルの共同研究とご指導を頂きました．鳥取大学山田茂教授には最尤法に関するご指導を頂きました．さらに，本論文のとりまとめに際して，ＮＴＴソフトウェア株式会社 高村真司取締役相談役，森道直取締役，細谷僚一取締役，福山峻一部長，吉田清部長からは，終始，暖かいご支援と励ましを頂きました．皆様方に心からお礼申し上げます．



# 文　献

## 第1章

[1] 亀田靖浩，”大規模ソフトウェア開発の現状と課題，”信学会誌，vol.74，no.5，pp.457-460，May 1991.

[2] 島崎恭一，”情報システムの現状と展望，”ＮＴＴ技術ジャーナル，pp.8-11，no.8，1995.

[3] S.E.HonIII,"Assuring software quality through measurements: A Buyer's perspective," J.Systems Soft., vol.13, pp.117-130, 1990.

[4] M.v.Genuchten,"Why is software late? An empirical study of reasons for delay in software development," IEEE Trans. Software Eng., vol.17, no.6, pp.582-590, June 1991.

[5] 中川豊，”ソフトウェア信頼度成長曲線の分析に基づく連結指数形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.J77-D-I，No.6，pp.433-442，June 1994.

[6] 中川豊，竹中市郎，”エラー複雑度に基づくソフトウェア信頼性モデル，”電子情報通信学会論文誌，Vol.J74-D-I，No.6，pp.379-386，June 1991.

[7] 中川豊，竹中市郎，”フォールト検出難易度に基づく残存フォールト推定モデルの適用法，”電子情報通信学会論文誌，Vol.J80-D-I，No.3，pp.282-290，March 1997.

[8] 伊土誠一，中川豊，杉村康，”運用データを利用した大規模オンライン情報処理システム用信頼性保証方式，”電子情報通信学会論文誌，Vol.J79-D-I，No.12，pp.1192-1202，Dec. 1991.

[9] 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.

[10] G.J.Shick and R.W.Wolverton,"An analysis of competing software reliability models," IEEE Trans. Software Eng., vol.SE-4, no.2, pp.104-120, March 1978.

[11] B.Littlewood and J.L.Verrall,"Likelihood function of a debugging model for computer software reliability," IEEE Trans. Rel., vol.R-30, no.2, pp.145-148, June 1981.

[12] E.Nelson,"Estimating software reliability from test data," Microelectronics and Reliability, vol.17, pp.67-74, 1978.

[13] J.R.Brown and M.Lipow,"Testing for software reliability," in Proc.Int. Conf. Reliable Soft., pp.518-527, April 1975.

[14] C.V.Ramamoorthy and F.B.Bastani,"Software reliability: Status and perspectives," IEEE Trans. Software Eng., vol.SE-8, no.4, pp.359-371, July 1982.

[15] J.D.Musa,"A theory of software reliability and its application," IEEE Trans.

Software Eng., vol.SE-1, no.3, pp312-327, Sep. 1975.
[16] A.L.Goel and K.Okumoto,"Time-dependent error-detection rate model for software reliability and other performance measures," IEEE Trans. Rel., vol.R-28, no.3, pp.206-211, Aug. 1979.
[17] J.D.Musa and K.Okumoto,"A logarithmic Poisson execution time model for software reliability measurement," Proc. 7th Int. Conf. Software Engineering, pp.230-238, 1984.
[18] S.Yamada, M.Ohba and S.Osaki,"S-shaped reliability growth modeling for software error detection," IEEE Trans. Rel., vol. R-32, no.5, PP.475-478, Dec. 1983.
[19]大場充，梶山昌之：" 習熟型ソフトウェア信頼度成長モデル，" 情処学会ソフトウェア工学研究会，vol.28-6，pp.31-36，Feb. 1983.
[20]岩佐博：" エラーの重要度およびエラー修正過程を考慮した複合ソフトウェア信頼度成長モデルの提案，" 情処学論，vol.29，no.5，PP.506-512，May 1988.
[21] N.Karunanithi, D.Whitly and Y.K.Malaiya, "Prediction of software reliability using connectionist models," IEEE Trans. Software Eng., vol.SE-18, no.7, PP.563-574, JULY 1992.
[22] Y.Tohma, K.Tokunaga, S.Nagase, Y.Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution," IEEE Trans. Software Eng., vol.SE-15, no.3, PP.345-355, March 1989.
[23] K.Matsumoto, K.Inoue, T.Kikuno and K.Torii,"Experimental evaluation of software reliability growth models", Proc. of 18-th International Symposium on Fault Tolerant Computing, pp.148-153, 1988.
[24]山田茂，" ソフトウェア信頼性評価技術，" ＨＢＪ出版局，1989.
[25] T.Furuyama, and Y.Nakagawa, "A Manifold Growth Model That Unifies Software Reliability Growth Models," International Journal of Reliability, Quality and Safety Engineering, vol.1,no.2, pp.161-184, 1994.
[26] J.W.Duran and J.J.Wiorkowski,"Capture-recapture sampling for estimating software error content," IEEE Trans. SoftwareEng., vol.SE-7, no.1, pp.147-148, Jan. 1981.
[27] 伊土誠一，" 「バグ捕獲・再捕獲」を用いたソフトウェア信頼性保証について，" 情処学論，vol.34，no,12，pp.2534-2542，Dec. 1993.
[28] P.Yip,"Estimating the number of errors in asystem using a Martingale approach," IEEE Trans. Rel., vol.44, no.2, pp.322-326, June 1995.
[29] R.J.Rubey, J.A.Dana, P.W.Biche,"Quantitative Aspects of Software Validation," IEEE Trans. Software Eng., vol.SE-1, no.2, pp.150-155 JUNE 1975.
[30] A.Endres, "An Analysis of Errors and Their Causes in System Programs," IEEE Trans. Software Eng., vol.SE-1, no.2, pp.140-149 JUNE 1975.
[31] N.F.Schneidewind & H.M.Hoffmann, "An Experiment in Software Error Data



Collection and Analysis," IEEE Trans. Software Eng., vol.SE-5, no.3, pp. 276-286 MAY 1979.
[32] J.B.Bowen, "Standard error classification to support software reliability assessment", AFIPS, Proceedings of the National Computer Conference, Vol.49, pp.697-705 1980.
[33] R.L.Glass, "Persistent Software Errors," IEEE Trans. Software Eng., vol.SE-7, no.2, pp.162-168 MAR.1981.
[34] V.R.Basili,B.T.Perricone, "Software errors and complexity: an empirical investigation," Communications of the ACM, Vol.27, No.1, pp.42-52, Jan. 1984.
[35] J.S.Collofello, L.B.Balcom, "A proposed causative software error classification scheme," AFIPS, Proceedings of the National Computer Conference, vol.54, pp.537-545 1985.
[36] T.Nakajo, H.Kume, "A case history analysis of software error cause-effect relationships," IEEE Trans. Software Eng., vol.17, no.8, pp.830-838 Aug, 1991.
[37] 宮本勲，" ソフトウェアエンジニアリングの現状と展望，" ＴＢＳ出版，1982.
[38] 中所武司，" ソフトウェアのテスト技法，" 情報処理，Vol.24，No.7，pp.842-852，1983.
[39] J.K.Chaar, M.C.Halliday, I.S.Bhandari, R.Chillarege, "In-process evaluation for software inspection and test," IEEE Trans. Software Eng., vol.19, no.11, pp.1055-1070, 1993.
[40] A.M.Adler and M.A.Gray, "TA Formalization of Myers cause-effect graphs for unit testing," ACM SIGSOFT software engineering Notes, vol.8, no.5,pp.24-32, 1983.
[41] 野木兼六，古川善吾，保田勝通，" ソフトウェアテスト項目作成支援システム，" 日立評論，vol.66，no.3，pp.29-32，1984.
[42] L.J.White and E.I.Cohen, "A domain strategy for computer program testing," IEEE Trans. Software Eng., vol.SE-6, no.3, pp.247-257, 1980.
[43] E.J.Weyuker and T.J.Qstrand, "Theories of program testing and application of revealing subdomains," IEEE Trans. Software Eng., vol.SE-6, no.3, pp.236-246, 1980.
[44] W.E.Howden, "Reliability of the path analysistesting strategy," IEEE Trans. Software Eng., vol.SE-2, no.3, pp.208-215, 1976.
[45] P.Piwowarski, M.Ohba, J.Caruso, "Coverage measurement experience during function test," Proc. of 15th ICSE, pp.287-299, 1993.
[46] D.Hamlet and R.Taylor, "Partition testing does not inspire confidence," IEEE Trans. Software Eng., vol.SE-16, no.12, pp.1402-1411, 1990.
[47] J.W.Duran and S.C.Ntafos, "An evaluation of random testing," IEEE Trans. Software Eng., vol.SE-10, no.4, pp.438-444, 1984.
[48] M.Z.Tsoukalas, J.W.Duran and S.C.Ntafos, "On some reliability estimation problems in random and partition testing," IEEE Trans. Software Eng., vol.19,

no.7, pp.687-697, 1993.
[49] T.Y.Chen and Y.T.Yu, "On the relationship between partition and random testing," IEEE Trans. Software Eng., vol.20, no.12, pp.977-980, 1994.
[50] T.Oka, T.Abe, S.Ido and J.Shindo, "Multi-terminal simulating test system," REVIEW of ECL, vol.23,no.9-10, pp.1100-1108, 1975.
[51] 拝原正人，神尾視教，柴武志，十倉健二，大町雄一，"多端末試験システム，"研実報，vol.28, no.12, pp.2725-2743, Dec. 1979.
[52] 斉藤珠喜，福岡允治，池田良一，長浜芳寛，戸田悟，萩原和夫，"DIPS 104-02システムサポートシステムの実用化，"研実報，vol.30, no.2, pp.361-383, 1981.
[53] F.D.Schulman, "Hardware measurement device for IBM system 360 time sharing evaluation," Proc. ACM, pp.103, 1967.

## 第2章

[1] A.L.Goel and K.Okumoto,"Time-dependent error-detection rate model for software reliability and other performance measures," IEEE Trans. Rel.,vol.R-28, no.3, pp.206-211, Aug. 1979.
[2] A.L.Goel, "Software Error Detection Model with Applications," J. Syst. and Softw., vol.1, pp.243-249, 1980.
[3] 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.
[4] J.D.Musa,"A theory of software reliability and its application," IEEE Trans. Software Eng., vol.SE-1, no.3, pp312-327, Sep. 1975.
[5] S.Yamada, M.Ohba and S.Osaki,"S-shaped reliability growth modeling for software error detection," IEEE Trans. Rel., vol. R-32, no.5, PP.475-478, Dec. 1983.
[6] 山田茂，大場充，"エラー発見率に基づくＳ字形ソフトウェア信頼度成長モデルの考察，"情処学論，vol.27, no.8, PP.821-828, 1986.
[7] 大場充，梶山昌之："習熟型ソフトウェア信頼度成長モデル，"情処学会ソフトウェア工学研究会，vol.28-6, pp.31-36, Feb. 1983.
[8] M.Ohba, "Software reliability analysis models," IBM J. RES. DEVELOP., vol.28, no.4, PP.428-443,JULY 1984.
[9] 岩佐博："エラーの重要度およびエラー修正過程を考慮した複合ソフトウェア信頼度成長モデルの提案，"情処学論，vol.29, no.5, PP.506-512, May 1988.
[10] 日本数学会（編）：岩波 数学辞典 第３版，岩波書店，pp.851-852, 1985.
[11] 中川豊，杉村康，内山公昭，"ＴＳＳにおけるデグレード試験手法，"昭５４信学情報・システム部門全大，p.379, 1979.



[12] 小野隆夫，鎌田正志："通信ソフトウェアの開発管理，" NTT R&D, vol.40, no.11, pp.1483-1490, 1991.
[13] 樋上重夫："ＹＰＳ適用による品質向上への取り組み，" 富士通ジャーナル，vol.18, no.2, PP.20-27, 1992.

## 第3章

[1] A.L.Goel, "Software Error Detection Model with Applications," J. Syst. and Softw., vol.1, pp.243-249, 1980.
[2] 山田茂，尾崎俊治，"ソフトウェアの信頼度成長モデルとその比較，"信学論，Vol.J65-D, no.7, pp.906-912, 1982.
[3] 三宅武，"ソフトウェアの品質評価法，"日科技連，1981.
[4] 山田茂，"ソフトウェア信頼性評価技術，"ＨＢＪ出版局，1989.

## 第4章

[1] 藤野喜一，花田収悦，"ソフトウェア生産技術，"電子通信学会，1985.
[2] R.Fairly," Software Engineering Concepts," McGRAW-HILL, 1985.
[3] P.N.Misra,"Software reliability analysis," IBM SYSTEM JOURNAL, vol.22, no.3, pp.262-270, 1983.
[4] 山田茂，"ソフトウェア信頼性評価技術，"ＨＢＪ出版局，1989.
[5] 三宅武，"ソフトウェアの品質評価法，"日科技連，1981.
[6] A.L.Goel, "Software Reliability Models: Assumtions, Limitations, and Applicability," IEEE Trans.Software Eng., vol.SE-11, no.12, pp.1411-1423,1985.

## 第6章

[1] T.Oka, T.Abe, S.Ido and J.Shindo, "Multi-terminal simulating test system," REVIEW of ECL, vol.23,no.9-10, pp.1100-1108, 1975.
[2] 拝原正人，神尾視教，柴武志，十倉健二，大町雄一，"多端末試験システム，"研実報，vol.28, no.12, pp.2725-2743, Dec. 1979.
[3] 斉藤珠喜，福岡允治，池田良一，長浜芳寛，戸田悟，萩原和夫，"DIPS 104-02システムサポートシステムの実用化，"研実報，vol.30, no.2, pp.361-383, 1981.
[4] 中川豊，杉村康，内山公昭，"ＴＳＳにおけるデグレード試験手法，"昭５４信学情報・システム部門全大，p.379, 1979.

# 索引

## ま行



## や行



## ら行



## わ行