■1 群（信号・システム）- 12 編（信頼性理論）

# 6 章 ソフトウェアの信頼性

（執筆者：井上真二，山田　茂）［2009 年 6 月 受領］

## ■概要■

ソフトウェアの信頼性は，使用期間後に物理現象としての摩耗・劣化によって，規定の機能を失うような故障発生プロセスをもつ一般的なハードウェア製品に関する信頼性の考え方とは異なる．ソフトウェア製品とは，ハードウェア製品とは異なり，ある規定された動作を実現するために考えられた論理の集まりをプログラムによって実現した知的生産物である．したがって，ソフトウェア製品の故障発生プロセスは，入力データとプログラム内部の状態にのみ依存する．つまり，ソフトウェア故障は，ソフトウェアの開発過程において意図せずに作り込まれたプログラム内の論理的な誤りや欠陥（ソフトウェアフォールト）によって発生し，一方，ソフトウェア動作環境においては，ソフトウェア故障を観測することによって，我々はプログラム内にその原因であるフォールトが潜在していることを認知するに至る．よって，ソフトウェアの信頼性は，ソフトウェアが所定の期間及び環境の下で，ソフトウェア故障が発生することなく動作する，もしくは機能やサービスを提供できる性質や度合と考えることができる．

また，ソフトウェアの信頼性は，標準的なソフトウェア品質を具体的に表現する六つの品質特性（機能性，信頼性，使用性，効率性，保全性，移植性）のうちの一つにもあげられており，上述した性質から「当たり前品質」として従来より取り扱われている．すなわち，ソフトウェア品質の観点から考えたとき，ソフトウェアの信頼性は，ソフトウェア開発過程において十分に確保しなければならない性質であり，それが不足する場合はソフトウェアとしての最低限の機能やサービスを提供できないような極めて重要な性質である．品質 3 要素（品質（Quality），費用（Cost），納期（Delivery））の調和を図りながら，ソフトウェアの品質／信頼性をその開発工程において確保するためには，各開発工程に対応した代表的な固有技術は去ることながら，各工程での中間成果物を審査するレビューやインスペクションなどのいわゆるソフトウェアフォールト混入予防技術や，定量的な信頼性評価技術に代表される管理技術を有効に利用することが重要である．

## 【本章の構成】

本章では，ソフトウェア信頼性概論（6-1 節）において，ソフトウェアの信頼性に関する諸問題に対処するために必要な基本的考え方について述べるとともに，ソフトウェア信頼性モデル（6-2 節）では，高信頼性ソフトウェアを実現するための管理技術として，定量的なソフトウェア信頼性評価技術に関する基礎的な理論について述べる．

■1 群 - 12 編 - 6 章

# 6-1 ソフトウェア信頼性概論

（執筆者：井上真二，山田　茂）［2009 年 6 月 受領］

現代の社会生活において，コンピュータシステムは主要な役割を担っており，重要な社会生活基盤として認知されている．その反面，一度コンピュータシステムに障害が発生すれば，我々の社会生活に多大な損害を与えることも少なくない．コンピュータシステム障害の原因としては，それを制御するソフトウェアの誤作動によるものが多い．したがって，コンピュータシステムの品質を向上させるためには，ソフトウェアの品質，特にその極めて重要な特性の一つである信頼性を向上させることが不可欠である．また，通信技術の急激な発展に伴うソフトウェア開発形態やソフトウェアサービス形態の変化など，ソフトウェアを取り巻く状況も時代の変遷と共に急激に変化しており，ソフトウェアの信頼性に関する諸問題は，今後も重要な問題として取り扱われるであろう．ここでは，ソフトウェアの信頼性をソフトウェア品質を構成する主要な特性の一つとしてとらえ，ソフトウェアの信頼性に関する諸問題に対処するために必要な基本的考え方について述べる．

## 6-1-1 ソフトウェア品質／信頼性

ソフトウェア品質（software quality）について考えるとき，それを構成する主要な特性，いわゆるソフトウェア品質特性（software quality characteristics）を把握しておく必要がある．国際的な規格としては，ISO/IEC 9126 によって定義されている六つの標準的なソフトウェア品質特性がある．表 6･1 には，その標準的なソフトウェア品質特性とそれらの定義から解釈できる簡単な意味についてまとめている．

**表 6･1　標準的なソフトウェア品質特性**

| 品質特性 | 意　味 |
|---|---|
| **機能性**（Functionality） | **規定された機能が備わっているかどうか．** |
| **信頼性**（Reliability） | **機能を期待通り実行できるかどうか．** |
| **使用性**（Usability） | **ユーザが運用・管理しやすいかどうか．** |
| **効率性**（Efficiency） | **資源や性能を有効に利用できているかどうか．** |
| **保守性**（Maintainability） | **仕様変更に容易に対応できるかどうか．** |
| **移植性**（Portability） | **異なった運用環境へ容易に移すことができるかどうか．** |

表 6･1 より，ソフトウェア品質における「信頼性」は，ソフトウェアとして最低限の機能やサービスを提供するために必要不可欠な品質特性であり，充足されて当たり前の品質，いわゆる「当たり前品質」ともいえる．表 6･1 にあげたソフトウェア品質特性には，更にそれらを具体的に説明する品質副特性（subcharacteristics）が存在する．特に，「信頼性」においては，「成熟性（maturity）」：ソフトウェア故障発生頻度はどの程度か，「障害許容性（fault tolerance）」：ソフトウェア故障が発生してもユーザへ迷惑をかけることがないか，「回復性（recoverability）」：ソフトウェア故障からの復旧がすばやく行えるか，という三つの品質副特性が存在する．

以上に示したソフトウェア品質／信頼性を満足したソフトウェア製品を開発するためには，それらをソフトウェア開発段階において十分に作り込む必要がある．近年，様々なソフトウェ

ア開発手法が提案され実用に供されているが，現在でも最も基本的かつ一般的な開発プロセスは，ウォーターフォールモデル（water-fall model）と呼ばれる開発プロセスである．これは，要求仕様定義→設計→コーディング→テストの各工程を経てソフトウェア製品が開発されるプロセスであり，テスト以外の各工程では，要求仕様書，基本・詳細設計書，ソースコードのような中間成果物が生成される．

一般的に，前述のような開発プロセスを経て開発されるソフトウェアシステムは，ある規定された動作を実現するために考えられた論理の集まりであり，開発過程において，論理的誤りや欠陥の混入を完全に防ぐことはできず，同時に，意図せず混入したそれらを完全に除去することも極めて困難であることが経験的に知られている．したがって，狭義の意味でソフトウェア信頼性（software reliability）を，「所定の環境及び期間において，ソフトウェア故障が発生することなく動作する，もしくは，機能やサービスを提供することができる性質や度合い」として考えることもできる．

ここで，ソフトウェアの信頼性に関する用語の定義を行っておく．ソフトウェアシステムとは，各開発工程において生成された中間成果物と製品としてのコンピュータプログラムの系を指し，ソフトウェアとは，一般的に，ソフトウェア製品としてのコンピュータプログラムを指す．ソフトウェア故障（software failure）とは，実行過程においてソフトウェアが期待どおりに動作せず，正しく動作せず機能しないことと定義される[1)]．また，ソフトウェアフォールト（software fault）とは，ソフトウェア故障の原因[2)]，すなわち，各工程において意図せずに作り込まれた論理的誤りや欠陥を指す[1)]．なお，欠陥とは，ユーザ要求を完全に網羅できていないことや誤解釈など第三者でも容易に認識できるソフトウェア故障の原因として取り扱われるときがある[2)]．

### 6-1-2 ソフトウェアの高信頼化技術

ソフトウェアの信頼性を向上させるための技術として，ソフトウェア信頼性技術と呼ばれる技術が従来より議論されており，主に固有技術と管理技術に分類される[1)]．

このとき，固有技術とはソフトウェアの信頼性に直接的影響を与えるような技術であり，フォールトアボイダンス技術やフォールトトレランス技術などがあげられる．フォールトアボイダンス技術とは，ソフトウェア開発過程におけるフォールトの混入を防ぐための技術であり，要求仕様化技術，設計技術，コーディング技術をはじめ，各工程での中間成果物を審査するレビューやインスペクションなどのいわゆるフォールト混入予防技術[3)]があげられる．一方，前述したようなフォールト混入及び除去に関する経験的知見より，取り除くことができなかったフォールトに起因するソフトウェア故障が発生したとしても，それを可能な限り許容するためのフォールトトレランス技術も存在する．その基本的な考え方は，マルチバージョンソフトウェア技法（multi-version software technique）と呼ばれる技法であり，独立した $N(\geq 2)$ チームが同一の仕様に従いソフトウェアを開発する技法である．その代表的な実装方法[1, 4)]としては，リカバリブロック（recovery block）や $N$ バージョンプログラミング（$N$-version programming）などの技法がある．

リカバリブロック技法（図 6･1 参照）とは，同一の入力に対するバージョンの出力結果が規定された許容範囲を満たすまで異なったバージョンを実行するような技法であり，すべてのバージョンにおいてその出力結果が許容範囲を満たさなかった場合，いわゆるシステム障

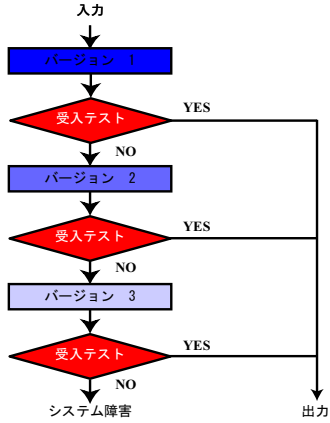


**図 6･1　リカバリブロック法の概念図（3 バージョンの場合）**

害が発生する．また，*N* バージョンプログラミング技法（図 6･2）とは，同一の入力に対する各バージョンからの出力結果をボーターと呼ばれる機能を介しながら，多数決論理に基づいた最終的な出力判定を行うような技法である．

一方，管理技術とは，ソフトウェア信頼性の定量的な計測・評価・予測技術，信頼性モデリング技術，データ収集技術など，ソフトウェアの信頼性に間接的影響を与える技術である．次節では，その中でも定量的な信頼性評価のための数理モデルであるソフトウェア信頼性モデルについて詳しく説明する．

また，コンピュータシステムの急激な発展と普及に伴い，ソフトウェア信頼性の重要性が高まる中，このソフトウェア信頼性技術を工学的観点から取り扱う学術分野は特に，ソフトウェア信頼性工学（software reliability engineering）と呼ばれている．

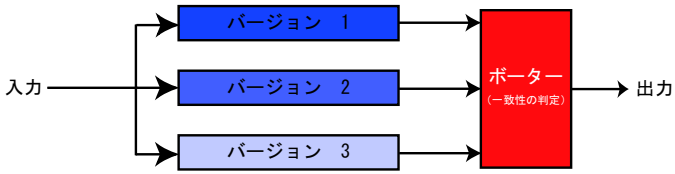


**図 6･2　*N* バージョンプログラミングの概念図（3 バージョンの場合）**

**■参考文献**

1) 山田茂，“ソフトウェア信頼性モデル－基礎と応用－,” 日科技連出版社，1994.
2) 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター，“定量的品質予測のススメ,” オーム社, 2008.
3) 森崎修司（編）, “ソフトウェアレビュー／ソフトウェアインスペクションと欠陥予防の現在,” 情報処理, vol.50, no.5, pp.375-417, 2009.
4) H. Pham, “Software Reliability,” Springer-Verlag, 2000.

■1 群 - 12 編 - 6 章

# 6-2 ソフトウェア信頼性モデル

（執筆者：井上真二，山田　茂）[2009 年 6 月 受領]

ソフトウェアの信頼性は，開発過程において十分に確保しなければならない性質であり，それが不足する場合はソフトウェアとしての最低限の機能やサービスを提供できない．品質 3 要素（品質（Quality），費用（Cost），納期（Delivery））の調和を図りながら，ソフトウェアの品質，とりわけ，信頼性をその開発工程において確保するためには，前節において議論したソフトウェア信頼性技術の中でも固有技術はさることながら，定量的な信頼性評価技術に代表される管理技術を有効に利用することが重要である．本節では，高信頼性ソフトウェアを実現するための管理技術の中でも，定量的なソフトウェア信頼性評価のための数理モデルであるソフトウェア信頼性モデルに関する基礎的な理論について述べる．

## 6-2-1 ソフトウェア信頼性モデル

ソフトウェア信頼性モデル（software reliability model）[1, 2, 3, 4]は，定量的なソフトウェア信頼性評価を目的とした数理モデルとして知られており，管理的側面からソフトウェアの高信頼化を支援する基盤技術として実用に供されている．ソフトウェア信頼性モデルを構築する際には，開発プロセスに関する要因（レビュー回数，レビュー工数密度など），プロダクトに関する要因（開発規模，ファンクションポイントなど），及び資源要因（テスト人員，テスト項目数など）を考慮しながら，シンプルでかつ精度良く信頼性評価が行えるモデル開発が要求される．

ソフトウェア信頼性モデルは，1970 年代から現在に至るまで数多くのモデルが提案されており，これらは，解析のために必要とするデータ，要求される解析結果，更に，取り扱う環境に基づいて，図 6･3 のような階層的分類が可能である[2, 3]．第 1 階層では，開発プロセスにおいて経験的もしくは実験的に収集されるデータを用いて，種々の信頼性特性との関連性を定式化するモデルを特に経験モデル（empirical model）として，それ以外のソフトウェア信頼性モデルを解析モデル（analytical）として分類している．続いて，第 2 階層において，解析モデルは，ソフトウェアの実行過程に依存した信頼性特性の時間的挙動を記述する動的モデル（dynamic model）と時間的挙動を考慮しない静的モデル（static model）に分類される．

ところで，ソフトウェアの実行過程において，ソフトウェア故障の累積観測数は時間の経過に依存して増加する．一方，その原因となるフォールトはデバッギング作業によって修正・除

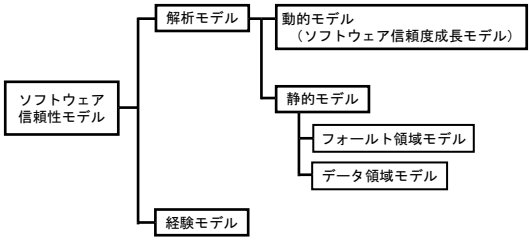


**図 6･3　ソフトウェア信頼性モデルの階層的分類[2, 3]**

去され，ソフトウェアの信頼性は向上し，ソフトウェア故障発生時間間隔は次第に長くなっていく．一般的に，このような現象をソフトウェア信頼度成長過程（software reliability growth process）と呼び，上述した「動的モデル」は，特に，ソフトウェア信頼度成長モデル（software reliability growth model: SRGM）[2, 3, 4]と称されることが多い．

### 6-2-2 ソフトウェア信頼度成長モデル

ソフトウェア信頼度成長モデルは，一般的に，費やされたテストもしくは運用時間や実行されたテストケース数などの時間的要因と発見フォールト数やソフトウェアハザードレートなどの信頼性特性との関係性を定式化した数理モデルともいえる．

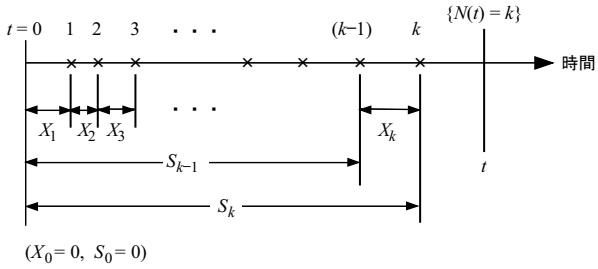


**図 6･4　ソフトウェア故障発生現象もしくはフォールト発見事象に関する確率諸量**

ソフトウェア信頼度成長モデルを構築するためには，ソフトウェア故障発生現象もしくはフォールト発見事象に関する確率諸量を定義する必要がある．図 6･4 には，ソフトウェア信頼度成長モデルを構築する際に取り扱う一般的な確率諸量を示している．図 6･4 に示した確率諸量は，それぞれ，次のように定義される：

$N(t)$ ＝時刻 $t$ までに発見された総フォールト数もしくは観測された総ソフトウェア故障発生数，

$S_k$ ＝ $k$ 番目のソフトウェア故障発生時刻 $(k = 0, 1, 2, \cdots ; S_0 = 0)$，

$X_k$ ＝ $(k-1)$ 番目と $k$ 番目のソフトウェア故障発生時間間隔 $(i = 1, 2, \cdots ; X_0 = 0)$．

なお，図 6･4 にある $\{N(t) = k\}$ は，時刻 $t$ までに $k$ 個のフォールトが発見されることを意味する．また，図 6･4 に示した確率諸量の定義から，$S_k = \sum_{i=1}^{k} X_i$，　$X_k = S_k - S_{k-1}$ がそれぞれ成立することも明らかである．

ここで，ソフトウェア故障は，その開発過程において意図せずに作り込まれたプログラム内のフォールトによって発生し，一方，ソフトウェア故障を観測することによって，我々はプログラム内にその原因であるフォールトが潜在していることを認知する．すなわち，図 6･4 に示した確率諸量では，フォールトが潜在しているプログラムパスが実行されることによって，ソフトウェア故障が観測される時間間隔を確率量として取り扱っている．また，これに伴い，$\{N(t), t \geq 0\}$ は経過時間に伴う累積発見フォールト数を表す確率過程（stochastic process）として一般的に取り扱われる．

ここで，時間的要因と発見フォールト数との関係性を定式化したモデルはフォールト発見数モデル（fault count model）と呼ばれ，ソフトウェアハザードレートとの関係性を定式化したものはハザードレートモデル（hazard rate model）と呼ばれる[2]．フォールト発見数モデルを信頼性評価に適用する場合は，フォールト発見数もしくはソフトウェア故障観測数を一定の時間ごとに数え上げた信頼性データが必要である．また，このようなモデルは，解析に必要なデータを比較的低い労力で収集できるため，実務においても多く適用されているモデルであり，これまで提案されているソフトウェア信頼度成長モデルの多くはこのタイプのモデルである．

**(1) 非同次ポアソン過程モデル**

フォールト発見数モデルの中でも，非同次ポアソン過程（nonhomogeneous Poisson process: NHPP）モデルは，比較的単純な構造をもち，適用性や妥当性の観点から実用に多く供されているモデルの一つである．特に，検出可能なフォールト数が有限個である場合を仮定した NHPP モデルは，一般的に次の三つの基本的仮定に基づいて構築される[5, 6]：

(1) ソフトウェア故障が発生した場合，その原因となるフォールトは直ちにかつ完全に修正・除去される
(2) 各ソフトウェア故障発生時刻は，それぞれ独立かつ同一の確率分布 $\Pr\{T \le t\} \equiv F(t)$ に従う
(3) テスト開始前にソフトウェア内に潜在する総フォールト数 $N_0 > 0$ は，有限であり，平均 $\omega(>0)$ のポアソン分布に従う

上に示した三つの基本的仮定から，時刻 $t$ までに $m$ 個のフォールトが発見される確率関数は，

$$\begin{aligned}\Pr\{N(t)=m\} &= \sum_n \binom{n}{m} \{F(t)\}^m \{1-F(t)\}^{n-m} \Pr\{N_0 = n\} \\ &= \sum_n \binom{n}{m} \{F(t)\}^m \{1-F(t)\}^{n-m} \frac{\omega^n}{n!} e^{-\omega} \\ &= \frac{\{\omega F(t)\}^m}{m!} \exp[-\omega F(t)] \qquad (m = 0, 1, 2, \cdots, n) \end{aligned} \tag{6･1}$$

のように求めることができる．ここで，式 (6･1) は，平均値関数が $M(t) = \omega F(t)$ である NHPP と本質的に等価であるため，ソフトウェア故障発生時間分布 $F(t)$ に対してある確率分布を適用することで，最終的に NHPP モデルを構築することができる．例えば，$F(t)$ に対して指数分布や 2 次のガンマ分布を適用した場合，それぞれ，代表的な NHPP モデルとしてよく知られている指数形ソフトウェア信頼度成長モデル[7]や遅延 S 字形ソフトウェア信頼度成長モデル[8]が得られる．また，実際のソフトウェア開発プロジェクトにおける最良のモデルを選択するための労力を削減することを目的として，一般化された確率分布を適用した一般化ソフトウェア信頼度成長モデルも提案されている[9]．

**(2) ソフトウェア信頼性評価尺度**

ソフトウェア信頼度成長モデルの適用手順[2]に従い，実際に観測された信頼性データを用いて，適用したモデルのパラメータ推定[2, 10]を行った後，様々な信頼性評価尺度に基づきな

がらソフトウェア信頼性の計測・評価・予測を行うことが重要である．

ソフトウェア信頼度（software reliability function）とは，時刻 $t$ までテストが進行しているとき，その後の時間区間 $(t, t+x](t \geq 0, x \geq 0)$ においてソフトウェア故障が発生しない確率として定義され，ソフトウェアの信頼性の考え方（本章 6-1 節参照）をモデル化したものとも考えることができる．定義より，ソフトウェア信頼度 $R(x \mid t)$ は，式 (6･1) から最終的に，$R(x \mid t) = \exp[-\{M(t+x) - M(t)\}]$ のように求められる．また，その他の代表的な尺度として，瞬間 MTBF（instantaneous MTBF）及び累積 MTBF（cumulative MTBF）がある．これらは特に，検出可能フォールト数が有限であることを仮定した NHPP モデルに対する平均ソフトウェア故障発生時間間隔（mean time between software failures）の代替的尺度として知られており，$m(t) \equiv dM(t)/dt$ とすると，それぞれ，$MTBF_I(t) = 1/m(t)$, $MTBF_C(t) = t/M(t)$ のように与えられる．

ソフトウェア信頼度成長モデルは，ここで議論したようなソフトウェア信頼性評価のための技術だけでなく，最適ソフトウェア出荷問題[11]をはじめとするソフトウェア開発管理面からの諸問題に関する議論にも幅広く応用されている．また，近年では，ソフトウェア信頼度成長過程がテスト時間要因のみでなく，テスト労力要因にも同時に依存するような，より現実的な仮定の下で議論される 2 変量ソフトウェア信頼度成長モデル[12, 13, 14]に関する議論もなされている．

**■参考文献**


1) J.D. Musa, A. Iannino and K. Okumoto, “Software Reliability: Mesurement, Prediction, Application,” McGraw-Hill, 1987.
2) 山田茂, “ソフトウェア信頼性モデル－基礎と応用－,” 日科技連出版社, 1994.
3) 山田茂，福島利彦, “品質指向ソフトウェアマネジメント,” 森北出版, 2007.
4) H. Pham, “Software Reliability,” Springer-Verlag, 2000.
5) N. Langberg and N.D. Singpurwalla, “Unification of some software reliability models,” SIAM J. Scie. Comp., vol.6, no.3, pp.781-790, 1985.
6) D.R. Miller, “Exponential order statistic models of software reliability growth,” IEEE Trans. Soft. Eng., vol.SE-12, no.1, pp.12-24, 1986.
7) A.L. Goel and K. Okumoto, “Time-dependent error-detection rate model for software reliability and other performance measures,” IEEE Trans Reliab., vol.R-28, no.3, pp.206-211, 1979.
8) S. Yamada, M. Ohba, and S. Osaki, “S-shaped reliability growth modeling for software error detection,” IEEE Trans. Reliab, vol.R-32, no.5, pp.475-478, 484, 1983.
9) 岡村寛之，安藤光昭，土肥正，“一般化ガンマソフトウェア信頼性モデル，” 電子情報通信学会論文誌, vol.J87-D-I, no.8, pp.805-814, 2007.
10) 岡村寛之，渡部保博，土肥正，尾崎俊治，“EM アルゴリズムに基づいたソフトウェア信頼性モデルの推定，” 電子情報通信学会論文誌, vol.J85-A, no.4, pp.442-450, 2002.
11) S. Yamada and S. Osaki, “Cost-reliability optimal release policies for software systems,” IEEE Trans. Reliab., vol.R-34, no.5, pp.422-424, 1985.
12) 石井智隆，土肥正，“二次元 NHPP に基づいたテスト労力依存型ソフトウェア信頼性モデル，” 電子情報通信学会論文誌, vol.J89-D, no.8, pp.1684-1694, 2006.
13) X. Cai and M.R. Lyu, “Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project,” Proc. 18th IEEE Intern. Symp. Soft. Reli. Eng., pp.17-26, 2007.
14) 井上真二，山田 茂，“2 変量ワイブル型ソフトウェア信頼度成長モデルとその適合性評価，” 情報処理学会論文誌, vol.49, no.8, pp.2851-2861, 2008.