

IT およびネットワーク

アプレット実行システムにおける信頼性

今 泉 充 啓
田 中 英 之
小谷野 錦 子

キーワード

アプレット (Applet)
システムコール (System call)
セキュリティ (Security)
オーバーヘッド (Overhead)
最適方策 (Optimal policy)

1. はじめに

最近, WWW (World Wide Web) 上でのアプリケーション構築を実現する技術の一つとして, アプレット (Applet) が注目されている。アプレットは, WWW において, プログラムコードをサーバからクライアントに動的にダウンロードし, クライアントのセキュリティ保全を行いながら実行することを可能としたプログラムコードである。代表的なアプレットとしては, Java アプレットがある。

アプレットはサーバからロードされるため, 不良コードによりシステム故障を引き起こすことがある。この問題に対処するため, 従来から様々な方策が提案されてきた^(1, 2, 3, 4, 5)。その一つにアプレットによるシステム資源への

アクセス（システムコール）をセキュリティチェッカにより監視する方策がある。セキュリティチェッカは、ユーザが指定したセキュリティ管理方針に基づいてシステムコールの検査を行い、必要に応じてプログラムコード実行の制限を行うことができるモジュールである⁽⁶⁾。システムコールの内容をチェックするセキュリティ検査は、検査プログラムを実行させることにより行われる。このセキュリティ検査により、不良コードによるシステム故障は未然に防ぐことができるが、この検査によるオーバヘッドは無視できない問題となっている。

セキュリティ検査によるオーバヘッド評価の問題は、既にいくつかの文献^(6,7,8,9,10)で紹介されている。しかし、その評価モデルの多くはシミュレーションによるものがほとんどであり、確率モデルとして定式化されていない。ここでは、アプレット実行システムを確率モデルとして記述し、種々の信頼性の問題を考察する。すなわち、アプレットがサーバからユーザ計算機にロードされたとき、JIT（Just-In-Time）コンパイラはユーザ計算機にダウンロードされ、これによりアプレットはオブジェクトコードに変換され実行が行われる。もし、サーバからの応答がない場合は、一定時間待ち再びアプレットのロードを要求する。

オブジェクトコードから行われるシステム資源へのアクセス（システムコール）はセキュリティチェッカにより監視する。セキュリティチェッカは、システムコール発行ごとに検査が必要なシステムコールかどうかを判断する。検査はある確率で行われ、この場合検査プログラムを実行する。セキュリティチェッカによる検査の結果、そのシステムコールはある確率で許可される。この場合、システムコールを再開してオペレーティングシステムに制御を渡す。逆にシステムコール不許可の場合は、エラーコードをアプレットに返し、アプレットの実行を中断する。中断後、不良コードの除去を行いアプレットの実行を再開する。セキュリティチェッカによる検査を行わなかった場合は、ある分布に従ってユーザーシステムへの不良アクセスによりシステム故障が発生するものと仮定する。この場合、ユーザーシステムの保全を

行い初期状態からやり直す。このとき、アプレット処理成功までの平均故障回数等を評価尺度として解析的に導出する。さらに、期待費用を求め、それを最小にする最適方策を議論し、最後に数値例による考察と評価を行う。

2. モデル

アプレット実行システムの概念図⁽⁶⁾を、図1に示す。

- (1) アプレットがサーバからクライアントにロードされたとき、JIT (Just-In-Time) コンパイラはクライアントにダウンロードされ、これによりアプレットはオブジェクトコードに変換され実行が行われる。アプレットのロードに要する時間は一般時間分布 $D(t)$ (平均 d) に従う。もし、サーバがビジー状態のため、サーバからの応答がない場合は、一定時間待ち再びアプレットのロードを要求する。この待ち時間は、一定時間分布 $W(t)$ (平均 w) に従う。また、アプレットの実行処理に要する時間は指数分布 $A(t)$ (平均 $1/a$) に従う。さらに、サーバは指数分布 $(1-e^{-\alpha t})(0 < \alpha < \infty)$ に従ってビジー状態になり、指数分布 $(1-e^{-\beta t})(0 < \beta < \infty)$ に従って正常状態に戻るとする。ここで、サーバの状態を、

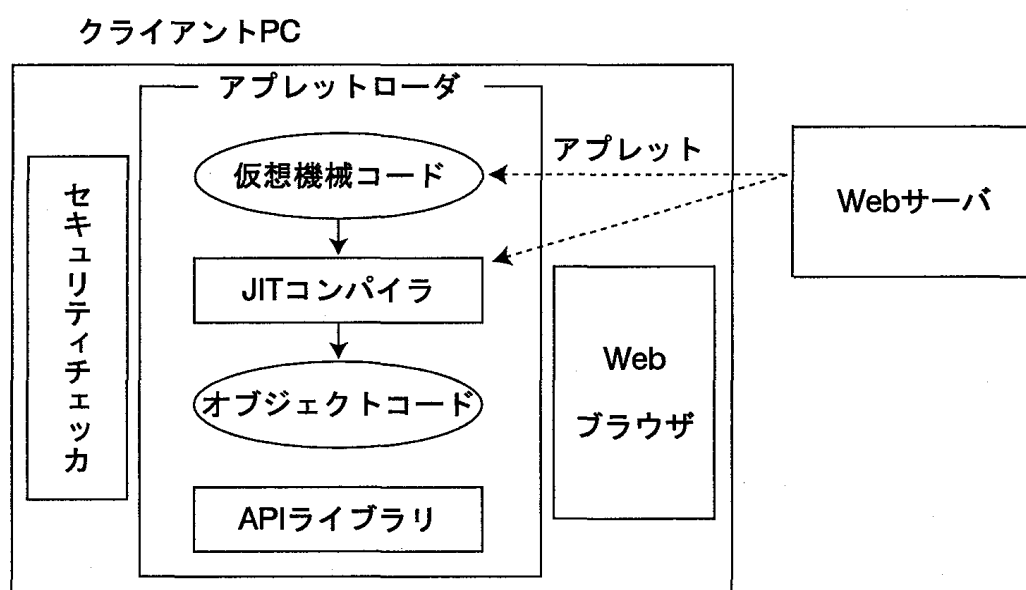


図1 アプレット実行システムの概念図

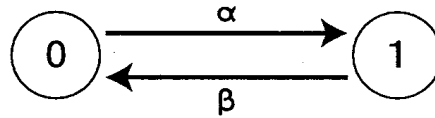


図2 サーバの状態推移図

状態0：正常状態。

状態1：ビジー状態。

と定義すると、各状態は、2状態をもつマルコフ再生過程⁽¹¹⁾を形成し、その推移は図2のように表される。

このとき、サーバが時刻0で状態*i*にあり、時刻*t*で状態*j*にある確率を $P_{ij}(t)$ ($i, j = 0, 1$) とおくと、 $P_{00}(0) = P_{11}(0) = 1, P_{01}(0) = P_{10}(0) = 0$ の初期条件のもとで、次のような状態確率を得る。

$$P_{00}(t) = \frac{\beta}{\alpha + \beta} + \frac{\alpha}{\alpha + \beta} e^{-(\alpha + \beta)t}, \quad (1)$$

$$P_{11}(t) = \frac{\beta}{\alpha + \beta} + \frac{\alpha}{\alpha + \beta} e^{-(\alpha + \beta)t}, \quad (2)$$

$$P_{01}(t) = 1 - P_{00}(t), \quad (3)$$

$$P_{10}(t) = 1 - P_{11}(t). \quad (4)$$

- (2) オブジェクトコードから行われるシステム資源へのアクセス（システムコール）はセキュリティチェッカにより監視する。システムコールは指数分布 $B(t)$ （平均 $1/b$ ）に従い発生する。
- (3) セキュリティチェッカは、システムコール発行ごとに検査が必要なシステムコールかどうかを判断する。検査は確率 p で行われ、この場合検査プログラムを実行する。検査プログラム実行に要する時間は一般分布 $U(t)$ （平均 u ）に従う。
- (4) セキュリティチェッカによる検査の結果、システムコールは確率 q で許可される。この場合、システムコールを再開してオペレーティングシステムに制御を渡す。システムコール処理に要する時間は、一般分布 $V(t)$ （平均 v ）に従う。

- (5) 逆に確率で $1 - q$ 不許可の場合は、エラーコードをアプレットに返し、アプレットの実行を中断する。中断後、不良コードの除去を行いアプレットの実行を再開する。中断後、再開に要する時間は一般時間分布 $G(t)$ (平均 μ) に従う。
- (6) セキュリティチェックにより、検査を行わなかった場合は、指数分布 $F(t)$ (平均 $1/\lambda$) に従ってユーザーシステムへの不良アクセスによりシステム故障が発生する。この場合、保全を行い初期状態からやり直す。この時間は、一般時間分布 $Z(t)$ (平均 z) に従う。

以上の仮定のもとで、システムの各状態を次のように定義する。

状態 0：アプレットロード開始。

状態 1：アプレット実行開始。

状態 2：システムコール発生。

状態 3：アプレット実行中断，不良コード除去開始。

状態 F：不良アクセスによるシステム故障発生。

状態 S：アプレット処理成功。

状態 E：アプレットロード待ち開始。

システムの状態を上のように定義すると、各状態は状態 S を吸収状態にもつマルコフ再生過程を形成し、各状態間の推移は図 3 のように表される。

マルコフ再生過程における 1 ステップ推移確率時間分布を $Q_{ij}(t) (i = 0, 1, 2, 3, E, F; j = 0, 1, 2, 3, E, F, S)$ とし、そのラプラス・スチルチェス (LS) 変換を $q_{ij}(s)$ とする。このとき、付録 1 により、次式を得る⁽¹²⁾。

$$q_{01}(s) = \int_0^{\infty} e^{-st} P_{00}(t) dD(t), \quad (5)$$

$$q_{0E}(s) = \int_0^{\infty} e^{-st} P_{01}(t) dD(t), \quad (6)$$

$$q_{EE}(s) = \int_0^{\infty} e^{-st} P_{11}(t) dD(t), \quad (7)$$

$$q_{E1}(s) = \int_0^{\infty} e^{-st} P_{10}(t) dW(t), \quad (8)$$

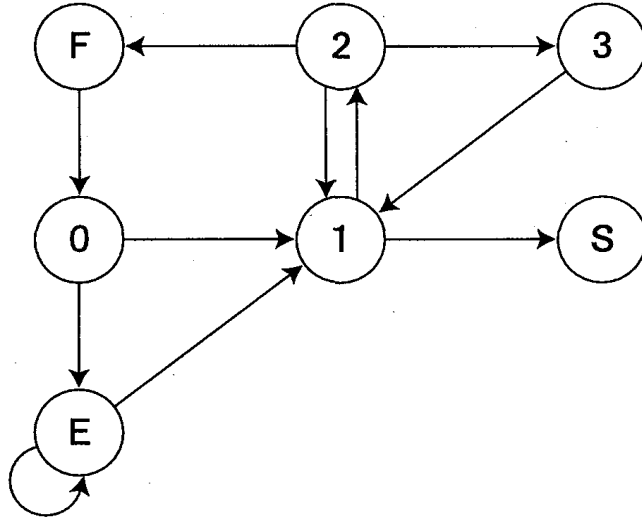


図3 システムの状態推移図

$$q_{12}(s) = \frac{b}{s + a + b}, \quad (9)$$

$$q_{1S}(s) = \frac{a}{s + a + b}, \quad (10)$$

$$q_{21}(s) = pqu(s)v(s) + (1-p)v(s+\lambda), \quad (11)$$

$$q_{23}(s) = p(1-q)u(s), \quad (12)$$

$$q_{2F}(s) = (1-p)\frac{\lambda}{s+\lambda}[1-v(s+\lambda)], \quad (13)$$

$$q_{31}(s) = g(s), \quad (14)$$

$$q_{F0}(s) = z(s). \quad (15)$$

最初に、アプレットロード開始からアプレット処理成功までの平均時間を求めよう。アプレット処理開始から処理成功およびシステム故障までの経過時間分布をそれぞれ $H_{1S}(t)$, $H_{1F}(t)$ とすると次式を得る。

$$H_{1S}(t) = \sum_{i=1}^{\infty} \{Q_{12}(t) * [Q_{21}(t) + Q_{23}(t) * Q_{31}(t)]\}^{(i-1)} * Q_{1S}(t), \quad (16)$$

$$H_{1F}(t) = \sum_{i=1}^{\infty} \{Q_{12}(t) * [Q_{21}(t) + Q_{23}(t) * Q_{31}(t)]\}^{(i-1)} * Q_{12}(t) * Q_{2F}(t). \quad (17)$$

ここで、一般に $\Phi^{(i)}(t)$ は分布関数 $\Phi(t)$ の i 重たたみこみを表す。すなわち、 $\Phi^{(i)}(t) \equiv \Phi^{(i-1)}(t) * \Phi(t)$, $\Phi_1(t) * \Phi_2(t) \equiv \int_0^t \Phi_2(t-u) d\Phi_1(u)$, $\Phi^{(0)}(t) \equiv 1$ である。よって、アプレットロード開始から処理成功までの経過時間分布を $H_{0S}(t)$ とすると、

$$H_{01}(t) \equiv Q_{01(t)} + Q_{0E}(t) * \sum_{i=1}^{\infty} Q_{EE}^{(i-1)}(t) * Q_{E1}(t), \quad (18)$$

と定義することによって,

$$H_{0S}(t) = \sum_{i=1}^{\infty} [H_{01}(t) * H_{1F}(t) * H_{F0}(t)]^{(i-1)} * H_{01}(t) * H_{1S}(t). \quad (19)$$

以上より,

$$\begin{aligned} h_{0S}(s) &\equiv \int_0^{\infty} e^{-st} dH_{0S}(t) \\ &= \frac{h_{01}(s)q_{1S}(s)}{1 - q_{12}(s)q_{2F}(s)q_{F0}(s)h_{01}(s) - q_{12}(s)q_{23}(s)q_{31}(s) - q_{12}(s)q_{21}(s)}, \end{aligned} \quad (20)$$

を得る。したがって、アプレットロード開始から処理成功までの平均時間 ℓ_{0S} を次のように求めることができる。

$$\begin{aligned} \ell_{0S} &\equiv \lim_{s \rightarrow 0} - \frac{dh_{0S}(s)}{ds} \\ &= \frac{wP_{01}(d) + dP_{10}(w)}{1 - P_{11}(w)} + \frac{1}{a} \\ &\quad + \frac{b}{a} \left\{ (1-p)[1 - v(\lambda)] \left[\frac{1}{\lambda} + z + \frac{wP_{01}(d) + dP_{10}(w)}{1 - P_{11}(w)} \right] + p[u + qv + (1-q)u] \right\}. \end{aligned} \quad (21)$$

次に、アプレット処理成功までの平均故障回数 M_F を求めよう。故障回数分布 $M_F(t)$ の LS 変換 $m_F(s)$ は次式で与えられる。

$$\begin{aligned} m_F &= \left\{ q_{01}(s) + q_{0E}(s) \sum_{i=1}^{\infty} [q_{EE}(s)]^{i-1} q_{E1}(s) \right\} \\ &\quad \sum_{j=1}^{\infty} \{ q_{12}(s)[q_{21}(s) + q_{23}(s)q_{31}(s)] \}^{j-1} q_{12}(s)q_{2F}(s)[1 + q_{F0}(s)m_F(s)]. \end{aligned} \quad (22)$$

よって、平均故障回数は,

$$M_F \equiv \lim_{s \rightarrow 0} m_F(s) = \frac{a}{b}(1-p)[1 - v(\lambda)]. \quad (23)$$

として求められる。

同様に、アプレット処理開始から処理成功またはシステム故障までの平均処理中断回数 M_3 は、

$$M_3 = \frac{bp(1-p)}{a+b(1-p)[1-v(\lambda)]}. \quad (24)$$

となる。

3. 最適方策

ここでは、経済性を考慮して、最適方策を議論する。システム故障に伴う損失費用を c_1 、中断に伴う損失費用を $c_2 (< c_1)$ 、システムの通常的な運用に伴う固定費用を $c_3 (< c_2)$ とし、アプレット処理成功までの期待費用 $C(p)$ を次のように定義する

$$\begin{aligned} C(p) &\equiv c_1 M_F + c_2 M_3 + c_3 \\ &= c_1 \frac{b}{a} (1-p)[1-v(\lambda)] + c_2 \frac{bp(1-q)}{a+b(1-p)[1-v(\lambda)]} + c_3. \end{aligned} \quad (25)$$

このとき、 $C(p)$ を最小にする最適なセキュリティ検査確率 p^* を求める。 $C'(p) = 0$ とおくと、

$$\frac{a(1-q)\{a+b[1-v(\lambda)]\}}{[1-v(\lambda)]\{a+b(1-p)[1-v(\lambda)]\}^2} = \frac{c_1}{c_2}, \quad (26)$$

を得る。ここで、式 (26) の左辺を $L(p)$ とおくと、次式を得る。

$$L'(p) = \frac{2ab(1-q)\{a+b[1-v(\lambda)]\}}{\{a+b(1-p)[1-v(\lambda)]\}^3} > 0, \quad (27)$$

$$L(0) = \frac{q(1-q)}{[1-v(\lambda)]\{a+b[1-v(\lambda)]\}}, \quad (28)$$

$$L(1) = \frac{(1-q)\{a+b[1-v(\lambda)]\}}{a[1-v(\lambda)]}. \quad (29)$$

よって、 $L(p)$ は $L(0)$ から $L(1)$ までの p の単調増加関数となる。以上から、

次のような結論を得ることができる。

- (i) もし, $L(0) < c_1/c_2 < L(1)$, すなわち, $a(1-q)/[1-v(\lambda)]\{a+b[1-v(\lambda)]\} < c_1/c_2 < (1-q)\{a+b[1-v(\lambda)]\}/a[1-v(\lambda)]$ ならば, 式 (26) を満たす有限で唯一の $(0 <) p^* (< 1)$ が存在する。
- (ii) もし, $L(0) \geq c_1/c_2$, すなわち, $c_1/c_2 \leq a(1-q)/[1-v(\lambda)]\{a+b[1-v(\lambda)]\}$ ならば, $p^* = 0$ である。この場合, セキュリティ検査を実施しないほうがよい。
- (iii) もし, $L(1) \leq c_1/c_2$, すなわち, $c_1/c_2 \leq (1-q)\{a+b[1-v(\lambda)]\}/a[1-v(\lambda)]$ ならば, $p^* = 1$ である。この場合, すべてのシステムコールを検査すべきである。

4. 数値例

3 章で求めた期待費用 $C(p)$ を最小にするセキュリティ検査確率 p^* について, 具体的な数値を求める。ここでは上述の解析で得られた方策についての大略の傾向, すなわち評価尺度の相対的な動向を把握することを主眼として, 平均システムコール処理時間 v をシステムの単位時間とおく。ここで, システムコール処理は指数分布に従うとする。また, アプレット実行処理に要する平均時間を $(1/a)/v = 1800$, 平均システムコール発生間隔を $(1/b)/v = 2$, 平均システム故障間隔を $(1/\lambda)/v = 300 \sim 1800$ (可変), システムコールを許可する確率を $q = 0.5 \sim 0.9$ (可変) とする。期待費用を求めるため, 中断に伴う損失費用 c_1 を単位費用とし, システム故障に伴う損失費用を $c_1/c_2 = 100 \sim 500$ (可変) と仮定する。

以上の仮定のもとで, 期待費用 $C(p)$ を最小にする p^* の数値例を表 1 に示す。例えば, $(1/\lambda)/v = 600$, $q = 0.7$, $c_1/c_2 = 300$ のとき, 最適なセキュリティ検査確率は $p^* = 0.85$ である。

表 1 $C(p)$ を最小にする最適セキュリティ検査確率 p^*

$(1/\lambda)/\nu$	q	c_1/c_2				
		100	200	30	400	500
300	0.5	0.51	0.75	0.86	0.92	1
	0.6	0.6	0.82	0.91	0.97	1
	0.7	0.7	0.89	1	1	1
	0.8	0.82	0.97	1	1	1
	0.9	0.97	1	1	1	1
600	0.5	0	0.37	0.61	0.75	0.85
	0.6	0.03	0.51	0.72	0.85	0.94
	0.7	0.25	0.67	0.85	0.96	1
	0.8	0.51	0.85	1	1	1
	0.9	0.85	1	1	1	1
1200	0.5	0	0	0	0.17	0.4
	0.6	0	0	0.1	0.4	0.6
	0.8	0	0.4	0.76	0.97	1
	0.8	0	0.4	0.76	0.97	1
	0.9	0.4	0.97	1	1	1
1800	0.5	0	0	0	0	0
	0.6	0	0	0	0	0.06
	0.7	0	0	0	0.15	0.45
	0.8	0	0	0.32	0.68	0.92
	0.9	0	0.68	1	1	1

表 1 によれば、最適セキュリティ検査確率 p^* は、 $(1/\lambda)/\nu$ の増大に伴い、減少する。また、 $q, c_1/c_2$ が大きくなるに従ってが増加する。ここで、 c_1/c_2 が大きくなるに従って p^* が増大するのは、システム故障に伴う損失が大きいときは、なるべくシステム故障に至らないようにセキュリティ検査確率を高く設定すべきであると解釈することができる。表 1 の結果から、 $(1/\lambda)/\nu$ が大きく、 q が小さい場合、 p^* は殆ど 0 となり、セキュリティ検査を実施しないほうがよいことがわかる。

5. おわりに

アプレット実行システムにおける高信頼化の問題を考察した。すなわち、サーバを含むアプレット実行システムの状態を考慮した新しい確率モデルを設定し、アプレット処理成功までの平均時間、平均故障回数、平均中断回数を求めた。さらに、経済性の観点から期待費用を導入し、それを最小にする最適方策について議論した。

数値例による考察から、期待費用を最小にする最適なセキュリティ検査確率は、システム故障発生間隔の増大に伴って減少し、システムコールを許可する確率、システム故障に伴う損失費用が大きくなるに従って増加する傾向を示した。

インターネットの進展とともに、オンライン電子商取引、オークションなどのアプリケーション構築を実現する技術として、アプレットの利用範囲は拡大している。このようなアプレット実行システムのセキュリティおよび信頼性評価の問題は、今後様々な視点からますます重要な課題になるものと考えられ、この方面に対する多くの研究が期待される。

参考文献

- (1) 大山恵弘, 加藤和彦, “SecurePot: システムコールフックを利用した安全なソフトウェア実行系”, 日本ソフトウェア科学会第 18 回大会論文集, 2001.
- (2) K. Kato and Y. Oyama, “SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation”, Technical Report ISE-TR-02-185, Institute of Information Sciences and Electronics, University of Tsukuba, January 2002.
- (3) J. Gosling and H. McGilton, “The Java Language Environment”, Sun Microsystems, 1995.
- (4) L. Gong, “Inside Java 2 Platform Security”, Addison Wesley Longman, 1999.
- (5) 阿部洋丈, 王維, シュラーニペーテル, 加藤和彦, “詳細なセキュリティポリシー記述のためのインタラクティブな作成環境”, 第 5 回プログラミングおよび応用システムに関するワークショップ, 2002.

- (6) 板橋一正, 松原克弥, 森山豊, 染谷祐一, 加藤和彦, 関口龍郎, 米澤明憲, “仮想機械独立なアプレットシステムの実現”, 信学論 (D-I), Vol.J84-D-I, No.6, pp.639-649, June 2001.
- (7) 品川高廣, 河野健二, 高橋雅彦, 益田隆司, “拡張コンポーネントのためのカーネルによる細粒度軽量保護ドメインの実現”, 情報処理学会論文誌, Vol.40, No.6, pp.2596-2606, June 1999.
- (8) 光来健一, 千葉滋, 益田隆司, “プロセスの依存関係に基づく分散システムのセキュリティ機構”, 第3回プログラミングおよび応用システムに関するワークショップ, 2000.
- (9) 品川高廣, 河野健二, 益田隆司, “実行可能コンテンツの安全な実行環境”, 情報処理学会論文誌, Vol.43, No.6, pp.1677-1689, June 2002.
- (10) 揚妻匡邦, 河野健二, 益田隆司, “モバイルコード技術を用いたアプリケーション層プロトコルの自動配布”, 第5回プログラミングおよび応用システムに関するワークショップ, 2002.
- (11) S. Osaki, “Applied Stochastic System Modeling”, Springer-Verlag Berlin, 1992.
- (12) K. Yasui, T. Nakagawa and H. Sandoh, “Reliability models in data communication systems, Stochastic Models in Reliability and Maintenance (edited by Osaki)”, pp.281-301 Springer-Verlag, Berlin, 2002.

付録

1. $Q_{i,j}(t)(i=0,1,2,3,E,F;j=0,1,2,3,E,F,S)$ の導出

システムが時刻 $t=0$ で状態 i から出発し、時刻 t までに次の状態 j へ推移する確率分布 $Q_{i,j}(t)$ とすると次式を得る。

$$Q_{01}(t) = \int_0^t P_{00}(t) dD(t), \quad (A.1)$$

$$Q_{0E}(t) = \int_0^t P_{01}(t) dD(t), \quad (A.2)$$

$$Q_{EE}(t) = \int_0^t P_{11}(t) dW(t), \quad (A.3)$$

$$Q_{E1}(t) = \int_0^t P_{10}(t) dW(t), \quad (A.4)$$

$$Q_{12}(t) = \int_0^t [1 - A(t)] dB(t), \quad (A.5)$$

$$Q_{1S}(t) = \int_0^t [1 - B(t)] dA(t), \quad (A.6)$$

$$Q_{21}(t) = pqU(t) * V(t) + (1-p) \int_0^t [1 - F(t)] dV(t), \quad (A.7)$$

$$Q_{23}(t) = p(1-q)U(t), \quad (A.8)$$

$$Q_{2F}(t) = (1-p) \int_0^t [1 - V(t)] dF(t), \quad (A.9)$$

$$Q_{31}(t) = G(t), \quad (A.10)$$

$$Q_{F0}(t) = Z(t). \quad (A.11)$$

ここで、* は分布関数のたたみこみを表す。