最適ソフトウェアアップデート戦略についての考察

B123456　広島大学　広大太郎　HIRODAI Taro
B234567　広島大学　広大花子　HIRODAI Hanako

1. はじめに

情報システムの社会における重要度は年々増加しており，システムの停止により多大なコストが発生することがある．近年では OpenStack などの仮想化基盤ソフトウェアを用いたシステムが企業を中心に盛んに用いられているため，そのような基盤ソフトウェアの信頼性・可用性の確保が重要な課題となっている．

システム停止の要因には障害によるものと保守によるものがある．一般に，ソフトウェアシステムの保守はソフトウェアをアップデートすることでソフトウェアに内在するバグや脆弱性を取り除き，システムの信頼性を向上させる．一方で，ソフトウェアアップデートではソフトウェアの機能のすべてあるいは一部を停止する必要がある．特に，基盤システムのソフトウェアアップデートはその上で稼働するアプリケーションに影響をおよぼすため，可用性の観点からアップデートに関する計画を行う必要がある．

ソフトウェアシステムの保守には開発者側とユーザ側の二つの視点がある．文献 [1] では，開発者がいつ最新版をリリースするかと言う問題について議論している．一方，文献 [2] ではユーザの視点から，いつアップデートと行うかと言う問題について議論している．文献 [2] では，ユーザがいつでも最新版が利用可能と仮定していた．これはオープンソースソフトウェアなどについては適用できるが，開発者が最新版のリリースを管理するようなソフトウェアではユーザがいつでも最新版を利用できる状況ではない．

そこで本稿では，ソフトウェア最新版のリリースが固定されているもとで，ユーザが「いつアップデートを行うか」と言う問題について議論する．特に，アップデートを行う費用とソフトウェアに内在するバグが引き起こす障害に関する費用から，システムのライフサイクルにおける総費用を最小にするような最適ソフトウェアアップデート戦略について考察する．

2. モデルの記述

時刻 $T_0 = 0$ においてユーザは最新版のソフトウェアの利用を開始し，時刻 $T_L(> T_0)$ までの利用を行う．開発者は期間 $[T_0, T_L]$ において N 回のソフトウェアリリースを行う．つまり，時刻列 $(T_0 \leq) T_1 < \cdots < T_N (\leq T_L)$ の時点で，そのときの最新版をリリースする．ユーザは期間 $[T_0, T_L]$ の任意時刻でソフトウェアアップデートを行うことができ，ソフトウェアアップデートによってソフトウェアをリリースされている最新の版にすることができる．

いま $D(t)$ を開発者が管理しているソフトウェア（開発版）に内在する残存バグ数を表す確率過程，$R(t)$ をリリースされたソフトウェア（リリース版）の残存バグ数を表す確率過程，$R(t)$ をユーザが利用しているソフトウェアの残存バグ数を表す確率過程とすると，開発者がリリースするタイミングにおいて，$D(T_i) = R(T_i),\ i = 1, \ldots, N$ の関係が成り立つ．また，時刻 t でユーザがソフトウェアアップデートを適用したとすると，$R(t) = U(t)$ の関係となる (図 1 を参照)．

ソフトウェアアップデートではシステムの停止やアップデート作業に関する費用がかかる．ここでは，一回のアップデートにつき $c_p\ (> 0)$ の費用がかかるものとする．一方，システムの障害は利用しているソフトウェアに内在する残存バグ数に比例した発生率 $\mu U(t)$ で起こるものと仮定する．システム障害が発生した場合，ユーザはそのときの最新版へアップデートするか，再起動などによる環境要因を変更することで対処する．本稿は，再起動などによる残存バグ数が変化しない対処を想定する．また，このとき単一の障害につき $c_f\ (> 0)$ の費用がかかるものとする．

3. 最適ソフトウェアアップデート戦略

前述の費用構造のもとでは，ユーザは必ず開発者が最新版をリリースした時点でアップデートを実行するかしないかを決定し，それ以外の時刻でユーザはアップデートを行わないことが容易に示される．つまり，問題は開発者が最新版をリリースする時刻列 $T_1, T_2, \ldots, T_N$ での動的計画方により最適なアップデート戦略の導出ができる．

いま，開発者が x 番目のリリースを行った時刻 T_x において，ユーザが $y(< x)$ 番目にリリースされたソフトウェアを利用している状況を考える．次の二つの費用を定義する．

- $W_P(x|y)$: x 番目にリリースされたソフトウェアへアップデートし，それ以降は最適なアップデート戦略を行ったときの時刻 T_L までの

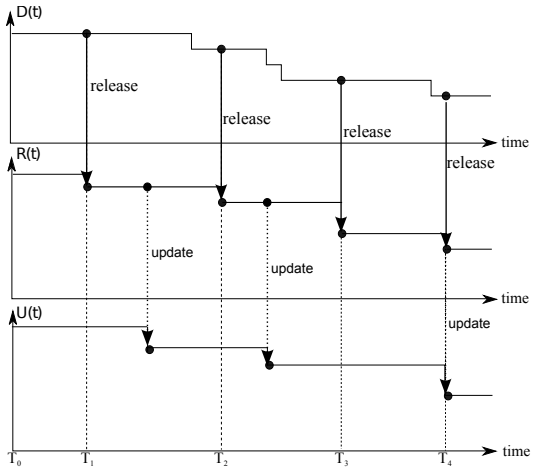

Figure 1: アップデートによる残存フォールト数のふるまい

総期待費用.

- $W_{\overline{P}}(x|y)$: x 番目にリリースされたソフトウェアへアップデートせずに，それ以降，最適なアップデート戦略を行ったときの時刻 T_L までの総期待費用.

このとき，最適性原理から $y = 0, \ldots, N$, $x = y+1, \ldots, N$ に対して以下の最適性方程式を得る.

$$V(x|y) = \min\{W_P(x|y), W_{\overline{P}}(x|y)\}, \quad (1)$$

$$W_P(x|y) = c_p + c_f\mu\Lambda(x|x) + V(x+1|x), \quad (2)$$

$$W_{\overline{P}}(x|y) = c_f\mu\Lambda(x|y) + V(x+1|y). \quad (3)$$

ここで

$$\Lambda(x|y) = (T_{x+1} - T_x)\mathrm{E}[R(T_y)] \quad (4)$$

である．さらに，$T_{N+1} = T_L$ および $V(N+1|\cdot) = 0$ であることに注意する．上記の最適性方程式は $y = N$, $x = N$ から後ろ向きに $V(x|y)$ を算出することができ，得られた $V(x|y)$ をもとにして，リリース時刻列 $T_1, \ldots, T_N$ でアップデートを行うかどうかの最適な行動が決定できる.

4. 数値例

開発版の残存バグ数が次の確率関数を持つ場合を考える.

$$P(D(t) = n) = \frac{(\omega\overline{F}(t))^n}{n!}e^{-\omega\overline{F}(t)}. \quad (5)$$

ここで，ω は時刻 $t = 0$ における総期待残存バグ数を表し，$F(t) = 1 - \overline{F}(t)$ はバグの発見時刻に関する確率分布を表す．数値例では，$\overline{F}(t)$ に関して次の二つの場合を考える.

Table 1: 総期待費用 (Case1)

λ	最適費用	適用なし	すべて適用
0.3	59.6	432.0	72.7
0.5	41.1	432.0	57.1
0.7	33.3	432.0	50.7

Table 2: 総期待費用 (Case2)

(α, β)	最適費用	適用なし	すべて適用
(1,18)	232.1	432.0	248.0
(3,18)	240.2	432.0	247.9
(5,18)	242.9	432.0	247.5

Case 1: $\overline{F}(t) = e^{-\lambda t}$.

Case 2: $\overline{F}(t) = 1/(1 + e^{-(t-\beta)/\alpha})$.

いま，$\omega = 100$, $T_L = 36$, $\mu = 0.12$, $c_p = 1.0$, $c_f = 1.0$ とし，開発者がソフトウェアをリリースする時刻列を 1.5 ずつの等間隔で $T_1 = 1.5, T_2 = 3.0, \ldots, T_{23} = 34.5$ と設定する．最適なアップデート戦略の効果を調べるため，アップデートを一回も行わない場合（適用なし）とすべてのリリース時刻でアップデートを行う場合（すべて適用）の総期待費用も導出する.

表 1，表 2 は残存バグ数が Case 1 と Case 2 の確率関数に従う場合の総期待費用を表している．表において,「最適費用」は 3 節の動的計画法によって導出された最適アップデート戦略を適用した場合の総期待費用を示している．表 1 と表 2 のどちらの場合においても，最適なアップデート戦略を行うことで期待総費用を低くすることができていることがわかる．特に，Case 1 の $\lambda = 0.7$ が最も効果が高く，すべてのリリース時刻でアップデートを行う場合の費用と比較して 34% の費用を削減することができた．$\lambda = 0.7$ は残存バグ数の減り方が最も急激な場合であり，そのような場合において最適なアップデート戦略の効果が高いことがわかる.

5. まとめと今後の課題

本稿では，ソフトウェアリリース時刻が固定されている環境でユーザ視点から「いつアップデートを行うか」という問題について議論した．今後は，ソフトウェア障害が発生したときに，最新版へのアップデートを行って修正する場合の最適アップデート戦略を考察する．また，セキュリティアップデートのように緊急で行われるものを考慮し，そのような環境下における最適なアップデート戦略について考察する予定である.

References

[1] H. Okamura, M. Tokuzane and T. Dohi, "Optimal security patch release timing under non-homogenous vulnerability-discovery processes", Proc. Int. Symp. on Software Reliab. Eng. (ISSRE'09), 120–128, 2009.

[2] C. Luo, H. Okamura and T. Dohi, "Optimal planning for open-source software updates", in submission.