# OSSプロジェクトデータに基づく統計的プロセス管理法とその応用に関する研究

山田　茂

キーワード：オープンソースソフトウェア (OSS)，管理図，ソフトウェア信頼度成長モデル (SRGM)，対数型ポアソン実行時間モデル，最適リリース問題

本稿は，山口 真和さんによる 2015 年度鳥取大学工学部に提出した卒業論文をもとに加筆修正したものです．

## 1. 問題の背景と論点

近年，次世代型ソフトウェア開発として注目されているオープンソースソフトウェア (OSS) の開発プロジェクトでは，ユーザが OSS を使用している間に何らかの不具合が発生すると，バグトラッキングシステムへ障害内容が登録される．その情報を開発者が確認し，ソースコードを改良することにより，新たなバージョンの OSS が公開されるという開発サイクルによって OSS プロジェクトの開発は進められる（図 1 参照）．つまり，OSS プロジェクトでは，従来の開発形態のような，フォールト（いわゆるバグと同義）の発見・修正・除去をテストチームにより行い，品質確認とともに品質／信頼性を評価するためのテスト工程が存在しない [1]．OSS 開発は，主にネットワーク上で複数の開発者が共同開発を行っており，開発速度が迅速であるという利点をもつ一方で，有効な品質管理手法が確立されておらず，OSS の信頼性に関する課題は多い．そこで本研究では，ソフトウェア信頼性評価モデルの一つである対数型ポアソン実行時間モデルを用いて，管理図による OSS 開発プロセスにおける品質の安定性の評価と開発期間の見積りが可能であることを示した．

## 2. ソフトウェア信頼度成長モデル

本研究では，非同次ポアソン過程 [2]（nonhomogeneous Poisson process，以下 NHPP と略す）に基づくソフトウェア信頼度成長モデルによる，OSS プロジェク

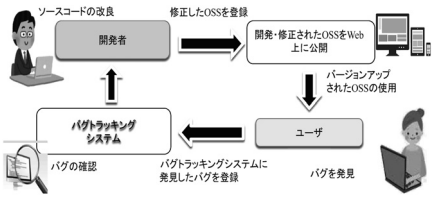


**図 1**　OSS プロジェクトにおける開発プロセス

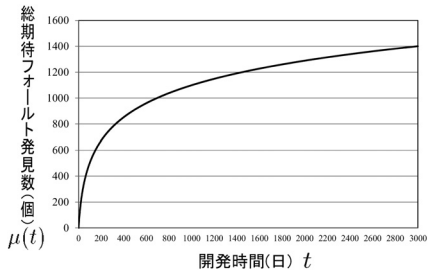


**図 2**　対数型ポアソン実行時間モデルの平均値関数

トの信頼性評価について議論する．検出可能フォールト数が無限であると仮定された NHPP に基づく，対数型ポアソン実行時間モデル [2, 3] の平均値関数 $\mu(t)$(時間区間 $(0, t]$ における，総期待フォールト発見数)，および強度関数 $h_\mu(t)$（単位開発時間当たりに検出されるフォールト数）は，それぞれ次式で与えられる．

$$\mu(t) = \ln(\lambda_0\theta t + 1)/\theta \ (\lambda_0 > 0, \theta > 0), \quad (1)$$

$$h_\mu(t) \equiv d\mu(t)/dt = \lambda_0/(\lambda_0\theta t + 1) = \lambda_0 e^{-\theta\mu(t)}. \quad (2)$$

また，式 (2) より，

$$\ln h_\mu(t) = \ln\lambda_0 - \theta\mu(t), \quad (3)$$

の関係を得る．ここで，$\lambda_0$ は初期故障強度，$\theta$ はソフトウェア故障 1 個当たりの故障強度の減少率を表す．また，ソフトウェア故障とは，ソフトウェアが期待どおりに動作しないことと定義される．また，ソフトウェア信頼性とは，ソフトウェア故障を引き起こすことなく動作することができる性質や度合いであり，これを確率で表現したものがソフトウェア信頼度である．図 2 は，実際の OSS プロジェクト (Android.1.5 NDK, Release 1) のバグトラッキングシステムのデータを用いてモデルパラメータ $\lambda_0$ および $\theta$ を推定したうえで，式 (1) の

やまだ　しげる
鳥取大学大学院 工学研究科社会経営工学講座
〒 680–8552 鳥取県鳥取市湖山町南 4–101
yamada@sse.tottori-u.ac.jp

  

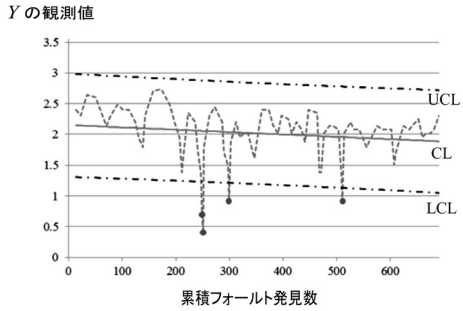


**図 3**　Android プロジェクトに対する管理図

平均値関数の推定結果を示したものである．

## 3. 管理図法

管理図とは，品質や製造工程が統計的に安定状態にあるかを判断するために使用するグラフである．ここでは，管理図の考え方を用いて，OSS 開発プロセスにおける品質の安定性を評価する．式 (3) において $Y = \ln h_\mu(t)$ とおくと，推定された $Y$ は次式となる．

$$\hat{Y} = \ln\hat{\lambda_0} - \hat{\theta}\hat{\mu}(t_G). \tag{4}$$

式 (4) の $t_G$ は所定のテスト時刻である．式 (4) は，プロジェクトが良好に進行し，フォールトの検出により安定的に信頼度成長が確保されているとき，強度関数の対数値は，総検出フォールト数の増加とともに，線形的に減少することを示している [2, 3]．したがって，式 (4) において $t = t_G$ で推定された対数値 $\hat{Y}_G = \ln\hat{h}_\mu(t_G)$ に対する $100(1-\alpha)\%$ 信頼区間は，回帰式の分散分析の結果を用いて次式により与えられる．

$$(\ln\hat{\lambda_0} - \hat{\theta}y_G) \pm t\left(n-2, 1-\frac{\alpha}{2}\right)\sqrt{V(\hat{Y}_G)}. \tag{5}$$

ここで，$y_G$ は $t = t_G$ のときの総発見フォールト数，$t(k,p)$ は自由度 $k$ の $t$ 分布の $100p\%$ 点，$V(\hat{Y}_G)$ は $\hat{Y}_G$ の分散である．式 (4) を中心線 (CL)，式 (5) を上方および下方管理限界線（UCL および LCL）として用いることで，管理図を作成できる．図 3 は，前出の Android 1.5 NDK プロジェクトデータを用いて作成した管理図である．OSS 開発プロジェクトが進行するにつれて，検出フォールト数が LCL を超えることもあったが，プロジェクトを通じて品質が安定化傾向を示していることがわかる．

## 4. OSS 最適リリース問題

本研究では，開発中の OSS の開発を終了して，実際の運用段階に移行するのに最適な開発期間を求める問題を OSS の最適リリース問題 [2] とする．OSS プロジェクトが目標フォールト発見率（ソフトウェア故障強度）$\lambda_F$ に到達する時間 $\tau_F$ は，次式で与えられる．

**表 1**　Android プロジェクトの最適リリース時刻

Android 1.5 NDK,Release 1　　$\lambda_0$ 8.596096　　$\theta$ 0.00038

| 前提条件 c1=1, c3=10, c4=1.0, Tu=500, λF=6.5 | | | | | |
|---|---|---|---|---|---|
| λ0 | c3/(c2-c1) | c2 | T0 | τF | T* |
| 8.596096 | 20 | 1.5 | | 104.5333 | 104.5333 |
| 8.596096 | 10 | 2 | | 104.5333 | 104.5333 |
| 8.596096 | 5 | 3 | 0.04612 | 104.5333 | 104.5333 |
| 8.596096 | 2.5 | 5 | 11.56111 | 104.5333 | 104.5333 |
| 8.596096 | 1.111111 | 10 | 30.18238 | 104.5333 | 104.5333 |
| 8.596096 | 0.714286 | 15 | 47.08825 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 20 | 62.68027 | 104.5333 | 104.5333 |
| 8.596096 | 0.416667 | 25 | 77.22413 | 104.5333 | 104.5333 |
| 8.596096 | 0.344828 | 30 | 90.90651 | 104.5333 | 104.5333 |
| 8.596096 | 0.294118 | 35 | 103.8645 | 104.5333 | 104.5333 |
| 8.596096 | 0.25641 | 40 | 116.2022 | 104.5333 | 116.2022 |
| 8.596096 | 0.227273 | 45 | 128.001 | 104.5333 | 128.001 |
| 8.596096 | 0.204082 | 50 | 139.3259 | 104.5333 | 139.3259 |

| 前提条件 c1=1, c2=40, c3=10, Tu=500, λF=6.5 | | | | | |
|---|---|---|---|---|---|
| λ0 | c3/(c2-c1) | c4 | T0 | τF | T* |
| 8.596096 | 0.526316 | 1 | 116.2022 | 104.5333 | 116.2022 |
| 8.596096 | 0.526316 | 1.5 | 59.03165 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 2 | 35.06446 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 2.5 | 22.93487 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 3 | 16.02387 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 3.5 | 11.74503 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 4 | 8.927041 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 4.5 | 6.980441 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 5 | 5.583783 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 5.5 | 4.550313 | 104.5333 | 104.5333 |
| 8.596096 | 0.526316 | 6 | 3.765799 | 104.5333 | 104.5333 |

$$\tau_F = \frac{1/\lambda_F - 1/\lambda_0}{\theta}. \tag{6}$$

次に，総期待ソフトウェアコストの定式化を行う．$c_1$ を開発段階のフォールト 1 個当たりの修正コスト，$c_2$ を運用段階のフォールト 1 個当たりの修正コスト，$c_3$ を単位開発時間当たりのコスト，$c_4$ を機会損失コスト係数と定義する．また，$T$ をリリース時刻，$T_U$ をメジャーバージョンアップまでの運用時間とする．このとき，総期待ソフトウェアコスト $C(T)(T > 0)$ は，式 (7) で与えられる．

$$C(T) = c_1\mu(T) + c_2\{\mu(T_U) - \mu(T)\} + c_3T + c_4T^2. \tag{7}$$

また，式 (7) の $C(T)$ を最小化する時間 $T = T_0$ は式 (8) で与えられる．

$$T_0 = \frac{-(2c_4 + c_3\lambda_0\theta) + \sqrt{(2c_4 - c_3\lambda_0\theta)^2 + 8c_4\lambda_0^2\theta(c_2 - c_1)}}{4c_4\lambda_0\theta}. \tag{8}$$

本研究では，信頼性要因として目標フォールト発見率を取り上げ，コストを最小化するにあたり，$T_0$ におけるフォールト発見率 $\lambda_p$ を目標フォールト発見率 $\lambda_F$ 以下にする要件も付加する．すなわち，OSS 最適リリース問題は，式 (9) として定式化される．

$$\left.\begin{array}{l}\text{Minimize}\ \ C(T) \quad \text{subject to}\ \ \lambda_P \le \lambda_F \\ \text{for}\ \ c_2 > c_1 > 0,\ c_3 > 0, c_4 > 0\end{array}\right\}. \tag{9}$$

$h_\mu(0) > c_3/(c_2 - c_1)$ のとき式 (9) を満たす最適リリース時期 $T^*$ は式 (10) で与えられる．

$$T^* = \max\{T_0, \tau_F\}. \tag{10}$$

95%信頼限界での目標ソフトウェア故障強度 $\tau_F$，ならびに式 (8) の $T_0$ を推定して，表 1 のように信頼性要件とコスト要因を考慮した最適リリース時刻 $T^*$ を算出することができた．

### 参考文献

[1] S. Yamada and Y. Tamura, *OSS Reliability Measurement and Assessment,* Springer-Verlag, 2016.
[2] 山田茂,『ソフトウェア信頼性の基礎―モデリングアプローチ―』, 共立出版, 2011.
[3] K. Okumoto, "A statistical method for software quality control," *IEEE Transactions on Software Engineering*, **SE-11**(12), pp. 1424–1430, 1985.