ソフトウェア信頼度成長モデルとベイズ統計機械学習によるオープンソースソフトウェア動的信頼性モデルの提案

杉山 透
放送大学
1520390580@campus.ouj.ac.jp

中谷 多哉子
放送大学
tinakatani@ouj.ac.jp

要旨

近年，産業界でオープンソースソフトウェア（OSS）を製品ソフトウェアに組み込んで使用することが増えてきているが，品質，特に信頼性に課題があると指摘されている．しかし，信頼性を評価するための既存のソフトウェア信頼性モデル，特に動的信頼性モデルは OSS 特有の開発形態と合っていないことが指摘されている．そこで，従来の動的信頼性モデルを取り込んだ状態空間モデルとベイズ統計機械学習による信頼性モデルを，従来の動的信頼性モデルに加えて OSS 特有の開発形態を考慮した動的信頼性モデルとして提案し，)実際の OSS の故障発生件数の予測を行った．提案手法を用いた予測結果を，従来の動的信頼性モデルにより同様の予測を行った結果と比較し，評価を行った．

1. 背景

近年，産業界におけるオープンソースソフトウェア（OSS）の開発・利用が一般的になっている．例えば，Anaconda,Inc. による Anaconda は，Jupyter Notebook のような OSS が多数組み込まれており，TypeFox 社の提供する gitpod には OSS である統合開発環境の theia が用いられている．

OSS の活用には，様々な利点が存在する．例えば，OSS が特定のベンダーに依存しない点，実装が公開されており独自に拡張することができる点等が挙げられる．しかし，OSS の品質が大きな課題となっている [8]．OSS の品質が社内基準を満たさない場合が多いためである．

システムやソフトウェアの品質について定めた ISO25010 において，ソフトウェア製品の品質は以下の 8 つの特性に分類されている [2].

1. Functional Suitability（機能適合性）
2. Performance Efficiency（性能効率性）
3. Compatibility（互換性）
4. Usability（使用性）
5. Reliability（信頼性）
6. Security（セキュリティ）
7. Maintainability（保守性）
8. Portability（移植性）

これらの品質特性の内，主に機能適合性や信頼性がソフトウェア製品に OSS を組み込む際に問題となると指摘されている [8]．このため，ソフトウェア製品に組み込む OSS の信頼性を作り込む必要がある．そして，信頼性を作り込んだ結果の評価に有用な手段がソフトウェア信頼性モデルである．

ソフトウェア信頼性モデルとは，開発プロセスや人的要因等を考慮して，ソフトウェア信頼性の定量的な評価を行う数理モデルである．ソフトウェア信頼性モデルには，大きく分けてテスト時間や運用時間等の時間的挙動を考慮せずに構築する静的ソフトウェア信頼性モデルと，時間的挙動を考慮して構築する動的ソフトウェア信頼性モデルの 2 つが存在する．しかし，OSS の信頼性モデル，中でも動的ソフトウェア信頼性モデルに基づいたものに関する研究はほとんど行われていない [11]．OSS の品質における特徴は，多くの開発者やユーザを巻き込んだ持続的なコミュニティを形成することで品質を向上さ

せていくことである [8]．このような特徴は，GitHub を始めとするソフトウェアの共同開発を支援するシステムが近年現れたことにより盛んになってきている．

対して，これまでの動的ソフトウェア信頼性モデルは，時間軸としてテスト項目の実施数を使うことがあるように，テストチームによるテスト工程を経ることが前提となっている．即ち，これまでのソフトウェア信頼性モデルと，OSS における信頼性の間にある乖離が大きくなってきている．そのため，OSS を自社製品に組み込むために信頼性を評価する場合，これまでのソフトウェア信頼性モデルでは正しく信頼性を評価できなくなる．この問題を解決するため，筆者らは状態空間モデルとベイズ統計機械学習を基本技術として用いた，OSS の信頼性を評価できる動的信頼性モデルを提案した [10]．その結果，従来の動的信頼性モデルが提案手法よりも高い精度で信頼性を評価できるという結果を得た．そこで，提案手法に従来の動的信頼性モデルを組み込むことで，提案手法の高精度化を試みた．本稿において，その結果を報告する．

本稿の構成を以下に述べる．まず 2 節にてソフトウェア信頼性モデルに関する関連研究を紹介し，関連研究における課題抽出を行う．3 節にて，モデル構築の基本技術について述べる．4 節では，公開されている実際の OSS バグレポートの集計結果と GitHub 上の開発履歴等を用いた信頼性モデルを実際に構築・評価し，従来の信頼性モデルによる評価結果との比較を行う．最後，5 節にて結果の考察や将来の展望を示す．

1.1. 用語の説明

本稿にて使用する用語の意味を表 1 に示す．

2. 関連研究

2.1. 本節で取り上げる関連研究について

ソフトウェアは，フォールトが見つかり，除去されることで信頼性が向上すると考えられる．信頼性の向上は信頼性の成長と言われ，定量評価手段として SRGM(software reliability growth model) が用いられている．SRGM は数多く提案されており，本論文では代表的な SRGM として以下 2 種を取り上げる．

1. 遅延 S 字型 SRGM
2. 対数ポアソン実行時間モデル

遅延 S 字型 SRGM は，モデルの意味が明白な代表的モデルであり [9]，筆者らによる研究で，OSS を対象としても高い精度で信頼度の成長を予測できた [10]．対数ポアソン実行時間モデルは，OSS の信頼度成長を表現する先行研究にて使用された実績 [11][14] がある．

加えて，状態空間モデルとベイズ統計機械学習を用いた信頼度推定を行った研究 [4][10] を示す．従来の SRGM は，故障件数をテスト時間や運用時間で説明するモデルであり，観測データのみで構成されている．対する当該研究においては，モデルに「状態」と呼ばれる潜在変数（状態変数）を考慮する．これにより，状態変数の時系列的変化に伴い故障件数も時系列的変化するモデルを表現することが可能になる．この状態変数の時系列的変化は，従来の SRGM よりも自由度の高いモデリングが可能であり，柔軟なモデルを構築できる．そして，状態空間モデルは構成要素として確率分布を用いる．そのため，データからモデルに用いられている確率分布のパラメータを推定する必要があり，推定の有効な手段となるのがベイズ統計機械学習である．

2.2. NHPP モデルによる SRGM

2.2.1 遅延 S 字型 SRGM

遅延 S 字型 SRGM（delayed S-shaped software reliability growth model）は，Yamada ら [13] が提案した SRGM である．本モデルは，故障の認知から，原因解析を経てフォールトの認知に至るまでのタイムラグを盛り込まれていることが特徴である．本モデルにおける平均値関数 $H(t)$ と強度関数 $h(t)$ を式 (1)(2) に示す．式中の a はその時点でソフトウェアに残存するフォールト数，b は 1 個あたりのフォールト発見率である．

$$H(t) = a[1 - (1 + bt)e^{-bt}] \quad (a > 0, b > 0) \tag{1}$$

$$h(t) = ab^2te^{-bt} \tag{2}$$

本モデルは，ソフトウェア内に存在する総フォールト数が有限であることが想定されている．しかし，OSS のような継続的に開発が行われていくようなソフトウェアでは，デバッグ時に新たなフォールトが混入する可能性があり，発見される総フォールト数は理論上無限大となると考える方が自然である．次に示す対数ポアソン実行時間モデルは，発見される総フォールト数が無限であるとしたモデルである．

表 1. 用語の意味一覧

用語	意味
ADVI	Automatic Differentiation Variational Inference，自動微分変分ベイズ
HMC 法	Hamiltonian Monte Carlo methods，ハミルトニアンモンテカルロ法
MCMC 法	Markov chain Monte Carlo methods，マルコフ連鎖モンテカルロ法
NHPP モデル	nonhomogeneous Poisson process モデル，ポアソン過程モデル
故障	アイテムが要求どおりに実行する能力を失うこと [7]
故障強度（瞬間故障強度）	修理アイテムの時間区間 $(t, \Delta t)$ での故障数を，その区間幅 Δt で除した値で，Δt を限りなくゼロに近付けたときの極限値 [7]
信頼性	アイテムが，与えられた条件の下で，与えられた期間，故障せずに，要求どおりに遂行できる能力 [7]
（ソフトウェア）フォールト	要求どおりに実行するのを妨げるソフトウェアアイテムの状態 [7]

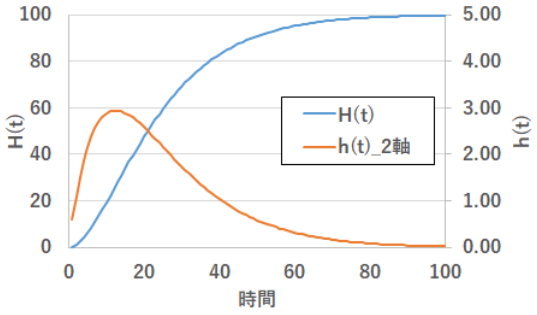

図 1. 遅延 S 字型 SRGM における信頼度成長（例）

2.2.2 対数ポアソン実行時間モデル

対数ポアソン実行時間モデル（logarithmic Poisson execution time model）は，Musa and Okumoto[6] が提案した SRGM である．本モデルは，強度関数 $h(t)$ が，実行時間（CPU 時間）までの平均値関数 $H(t)$ に関して指数関数的に減少するとしている．また，先述の通り，総フォールト数の上限を想定しておらず，テストをし続ける限り，無限にフォールトが発見されることになる．本モデルにおける平均値関数 $H(t)$ と強度関数 $h(t)$ を式 (3)(4) に示す．式中の λ_0 は初期故障強度，θ はソフトウェア故障 1 個あたりの故障強度減少率である．加えて，対数ポアソン実行時間モデルに則った信頼度成長の例を，図 2 に示す．

$$H(t) = \frac{1}{\theta}\ln(\lambda_0\theta t + 1) \quad (\lambda_0 > 0, \theta > 0) \tag{3}$$

$$h(t) = \frac{\lambda_0}{\lambda_0\theta t + 1} \tag{4}$$

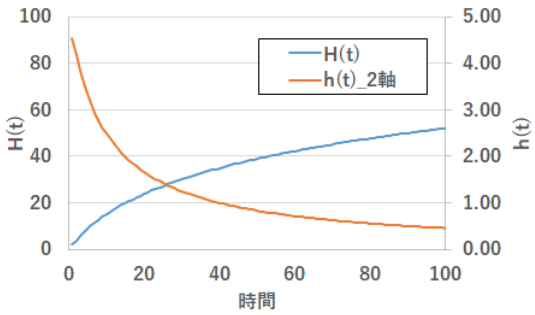

図 2. 対数ポアソン実行時間モデルにおける信頼度成長（例）

2.3. 状態空間モデルとベイズ統計機械学習による故障強度推定

2.3.1 カルマンフィルタによるアプローチ

貝瀬 [4] は，ソフトウェアの故障発生履歴から故障強度を推定する研究を行った．ソフトウェアを含むシステムの故障強度は，運用開始から時間経過と共に傾向が変化する．この傾向によって，システムの状態は初期故障

期間（故障率・故障強度減衰期間），偶発故障期間（故障率・故障強度一定期間），摩耗故障期間（故障率・故障強度増加期間）の 3 つの期間に分類される．分類法は，故障発生履歴から式 (5) に示すワイブル分布を推定し，パラメータ β の値によって判断するものが広く知られている．

$$f(t|\alpha,\beta) = \frac{\beta}{\alpha}(\frac{t}{\alpha})^{\beta-1}\exp[-(\frac{t}{\alpha})^{\beta-1}] \quad (\alpha>0,\beta>0) \tag{5}$$

式中の t は運用開始からの経過時間，α は尺度パラメータ，β は形状パラメータである．

貝瀬も，故障強度推定と同時にワイブル分布のパラメータを推定することで，ソフトウェアが各時刻においてどの期間の故障傾向を示すか推定した．当該研究では，ワイブル分布のパラメータが時間変化するモデルを構築し，ベイズ推定のデータからのパラメータ推定に長けた性質を利用して，カルマンフィルタによるパラメータのベイズ推定を行った．モデルの評価を行った結果，ワイブル分布のパラメータが時間変化するモデルは，パラメータが時間変化しないモデルよりもデータによくフィットする結果を得た．

当該研究は，観測データに含まれない潜在変数を柔軟に設定し，モデルに盛り込むのが特徴である．当該技術は，次節に示す筆者らの研究においても参考にした．

2.3.2 OSS 開発プロジェクトから抽出したデータを活用したアプローチ

筆者らは，OSS 開発プロジェクトの Git から抽出したデータを，様々な潜在変数と組み合わせて盛り込んだモデルを，OSS の信頼性評価に相応しいモデルとして提案した [10]．抽出・利用したデータは，コミットとアップデートの履歴である．筆者らは，これらを，1 章で示した OSS の特徴であるコミュニティにおける活動で信頼性を含む) 品質を高めた履歴であると捉えた．提案したモデルは，以下の 2 つである．前者を線形トレンドモデル，後者を定数係数モデルと呼称している．

以下に示すのが，線形トレンドモデルである．本モデルは，OSS の対数故障強度（$loglambda$）はトレンド要素（$trend$）と季節要素（$season$），コミット履歴（$C_{[t]}$）を定数倍したコミット要素（$commit$），アップデート履歴（$U_{[t]}$）を定数倍したアップデート要素（$update$）から成り立つとしたものである．本モデルのトレンド要素は，2 階線形トレンド要素として広く知られるものであり，トレンドがゆっくり滑らかに変化することを仮定したものである．この性質が SRGM のような滑らかな曲線を描くのに適していると考えた．

$$\begin{aligned}
trend_{[t]} &= \mathrm{Norm}(2*trend_{t-1} - trend_{t-2}, \sigma_{sys}) \\
\sum_{l=0}^{6} season_{[t-l]} &= \mathrm{Norm}(0, \sigma_{season}) \\
commit_{[t]} &= b_1 * C_{[t]} \\
update_{[t]} &= b_2 * U_{[t]} \\
loglambda_{[t]} &= trend_{[t]} + season_{[t]} + commit_{[t]} \\
&\quad + update_{[t]} \\
Y_{[t]} &= \mathrm{Poisson}(\exp(loglambda_{[t]}))
\end{aligned} \tag{6}$$

また，以下に示すのが定数係数モデルである．本モデルは，線形トレンドモデルのトレンド要素を変更したものである．モデル中の C_1 は定数係数であり，本係数が 0 より大きく 1 未満と推定されれば，トレンドが時事刻々と減衰し，SRGM のような曲線となると考えた．

$$trend_{[t]} = \mathrm{Norm}(C_1 * trend_{t-1}, \sigma_{sys}) \tag{7}$$

しかし，線形トレンドモデルは計算が発散し，定数係数モデルは故障発生予測精度が SRGM を下回った．

2.4. 関連研究における課題

OSS の信頼性評価について，動的信頼性モデルを用いた信頼性評価に関する研究がほとんど行われておらず，傾向分析に基づく事例研究としての文献はいくつか提案されているものの，これらは OSS のもつ特有の開発形態を考慮した信頼性評価を提案したものではないと指摘されている [11]．OSS 信頼性評価に関する研究として，田村ら [11]，山田 [14] による研究では，総フォールト数が無限となることを許容する対数ポアソン実行時間モデルが，OSS の開発形態を考慮した SRGM であると提案している．また，田村らは OSS 特有の要素であり信頼性を評価する際に重要な要素として，フォールト発見事象の複雑性やバグフィックス及びコンポーネントの加除が常に行われていることによるソフトウェア故障強度の変化を挙げている [11]．バグフィックスの履歴等は，近年は Git のコミット履歴といった形で詳細なデータに容易にアクセスできることが多く，モデルにも積極的に取

り込める環境が整ってきている．このようなデータを活かすには，先述の関連研究で用いられている，故障履歴と故障が発生した時間といった観測データ同士を直接結びつけるアプローチをさらに発展させ，様々なデータを取り入れることができるモデルを用いる必要がある．

貝瀬の研究 [4] では，状態空間モデルを構築し，モデル中の確率分布のパラメータをデータを用いベイズ統計機械学習するアプローチを取っている．これは，状態として OSS 特有の開発形態を盛り込むことができるアプローチであると考えられる．当該研究においても，故障発生間隔がワイブル分布に従うと考え，ワイブル分布のパラメータをデータからカルマンフィルタで逐次推定している．この手法により，例えばリリース直後の OSS の故障を除去することで，初期故障期間から偶発故障期間へ推移する様子を推定することが可能になる．しかし，カルマンフィルタは正規分布によって構築されたモデルにのみ適用可能な手法である．また，近年は理論とソフトウェアの進歩により，さらに複雑なモデルを構築し，ベイズ統計機械学習を適用することがことが可能になってきており，Git 上のデータもモデルに用いることができる．

コミット履歴のような，OSS 開発における様々なデータを柔軟に取り込み，OSS の信頼性評価に相応しいモデルとして筆者らが提案したのが，先述の線形トレンドモデルと定数係数トレンドモデルである [10]．しかし，故障発生予測精度が SRGM を下回る結果となってしまった．そこで，原因を精査し，さらに SRGM をトレンド要素として取り込むことで，提案手法の精度を向上させる．

筆者らの手法は，SRGM を状態空間モデルの枠組みで捉えてモデルの一部とし，ベイズ統計機械学習でパラメータ推定を行うものである．これら 2 つの技術は，筆者らの提案手法における基本技術である．次章にて，2 つの基礎技術について述べる．

3. 本稿で用いる基礎技術

3.1. 状態空間モデルについて

状態空間モデルは時系列解析において広く用いられるモデルであり，データ点同士が互いに関係を持つデータを確率的にとらえることができる．従来の SRGM では，観測値同士（例：故障発生時間と累積故障発生件数）の関係を直接記述するモデルを作成していた．対して，状態空間モデルでは，状態変数という潜在変数を導入したことにより，解釈に都合が良い状態変数を複数組み合わせて複雑なモデルを構築することが容易になった [3]．OSS の信頼性モデルを状態空間モデルで構築する場合，ソフトウェアのコミットのアップデート履歴データといった，OSS 特有の開発形態によりもたらされる豊富なデータを状態変数として用いることができる．

例として状態空間モデルの枠組みでソフトウェア信頼性モデルをとらえることを考える．この場合，観測量が時点 t における故障発生件数，状態が故障強度と考えることができる．さらに，ある時点 t における故障強度 h_t は，1 ステップ前の故障強度 h_{t-1} にのみ依存し，t における故障発生件数 Y_t は，t における故障強度 h_t によってのみ決まることになる．これを式で表現すると以下のようになる．このような，状態空間モデルの構造を表現する代表的な手法として，グラフィカルモデルを用いた表現と方程式を用いた表現の 2 つが存在する．グラフィカルモデルによる表現例を，図 3 に示す．状態である h_t が，観測量である Y_t の背後で変化している様子が表現されている．

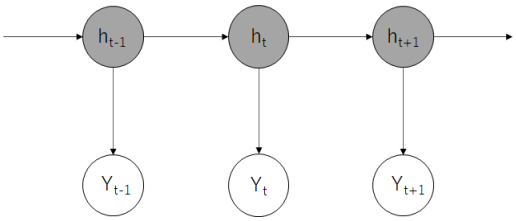

図 3. 状態空間モデルのグラフィカルモデル表現例

また，同じモデルを方程式で表現した例を，式 (8)(9) に示す．

$$h(t+1) = \mathrm{F}(h(t)) \tag{8}$$

$$Y(t) = \mathrm{G}(h(t)) \tag{9}$$

状態である h_t が状態方程式において関数 F に従って推移し，観測量である Y_t は，観測方程式において状態が関数 G に入力されることで生成されている様子が表現されている．

3.2. ベイズ統計機械学習について

ベイズ統計機械学習は，式 (10) のベイズの定理を基礎としている．式 (10) にて，x はデータ，θ は確率分布

のパラメータを指す.

$$p(\theta \mid x) = \frac{p(x \mid \theta)\, p(\theta)}{p(x)} \propto p(x \mid \theta)\, p(\theta) \qquad (10)$$

各項と文字の意味を以下に示す.

- x：観測値
- θ：モデル内における確率分布のパラメータ
- $p(\theta)$：事前分布．観測値 x を得る前のパラメータ θ の分布.
- $p(x \mid \theta)$：尤度．パラメータ θ のときデータ x が生成される確率
- $p(\theta \mid x)$：事後分布．観測値 x を得たときの θ の分布.
- $p(x)$：事後分布の面積を 1 にする正規化定数とみなせる.

上式は，モデルにおけるパラメータをデータにより更新する式と解釈することができる．しかし，パラメータが多い高次元モデルとなると，正規化定数 $p(x)$ の計算が難しくなることが知られている.

そこで，近年計算ベイズ統計でよく利用されている MCMC 法では，正規化定数 $p(x)$ を無視し，事後分布に比例する分布 $p(x \mid \theta)p(\theta)$ から乱数を大量に生成して，得られた分布から事後分布を作る．また，ADVI は，変分法を用いることで，近似解ではあるが MCMC 法よりも高速に事後分布を得ることができる.

4. ケーススタディ

4.1. モデル作成方針

先述の基本技術を用いて SRGM を取り込んだソフトウェア信頼性モデルを構築し，有効性を実際の OSS におけるデータを用いたケーススタディで検証する．モデル構築のインプットとして，OSS の故障履歴データ，コミット履歴を用いる．2.3.2 項における筆者らの研究 [10] では，アップデートの履歴も使用した．しかし，アップデートの履歴はコミットの履歴に内包されるため，結果に悪影響を及ぼす可能性があると考え，今回は使用していない．故障履歴データを分析することにより，故障のトレンド傾向や故障の周期的傾向の有無を抽出することができる．故障にトレンド傾向や周期性が見られた場合は，非定常データで過去に依存する傾向を持つデータであり，対応する要素をモデルに組み込む.

4.2. 推定と予測に用いるデータ

Eclipse のプラグインとして提供されている OSS である，JDT（Java development tools）の故障データを使用した．バグレポート 1 件を故障 1 件とし，起票日を故障発生日とした．また，開発チームの Git より，コミット履歴データを抽出した．抽出対象は，全てのコミットの日付と回数とした.

データを取得した期間を，表 2 に，JDT の故障発生件数の履歴を図 4 に示す.

表 2. JDT の各データ取得期間

データ	期間	日数（日）
故障データ	2001/10/10〜2013/11/19	4424
コミット履歴	2002/05/09〜2013/11/19	4213

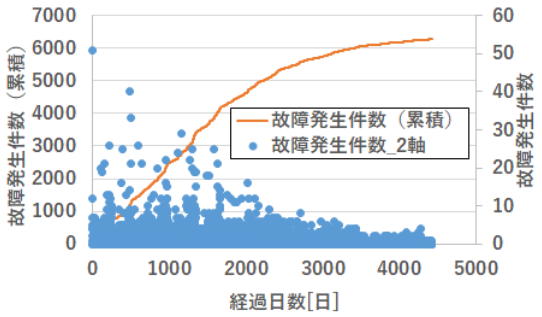

図 4. JDT の故障発生履歴

累積故障発生件数の上昇が，日数の経過ごとに鈍くなっているため，トレンドを持つデータであることがわかる．また，JDT の自己相関係数によるコレログラムを図 5 に示す．図中の青線が，自己相関の有無を判定する閾値である．図 5 より，概ね 7 日間で高い自己相関を示していることがわかり，7 日周期を表現する要素をモデルに組み込む必要がある.

さらに，JDT の偏自己相関係数によるコレログラムを図 6 に示す．自己相関係数によるコレログラムは，t_i 日目のデータと t_{i+a} 日目のデータの間の関係を調べようとする時，t_{i+1} 日目から t_{i+a-1} 日目までのデータの影響を受ける．しかし，偏自己相関係数によるコレログラムは，t_i 日目のデータと t_{i+a} 日目のデータの関係だ

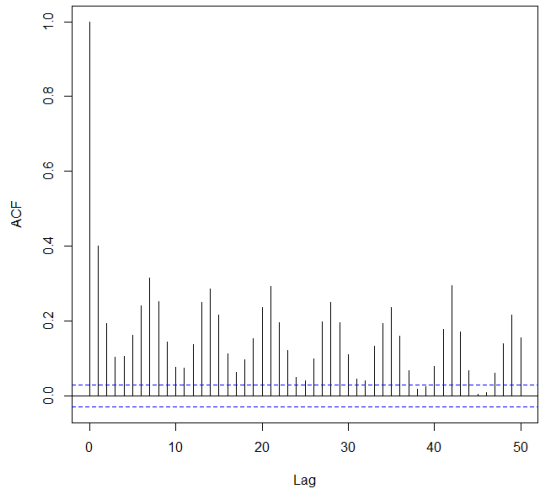

図 5. コレログラム˙50 日分（横軸：[日]）

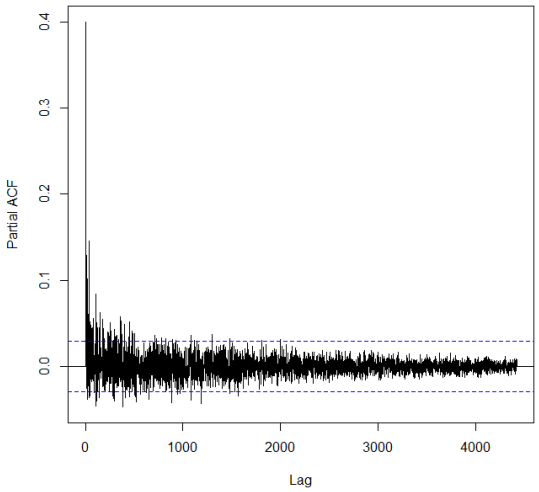

図 6. コレログラム（偏自己相関係数）（横軸：[日]）

けを純粋に調べることができる．よって，トレンドの効果を完全に除去してデータを評価できると考えた．

図 5 と同じく，図中の青線が，自己相関の有無を判定する閾値である．高い自己相関を示すのは，図 5 と同じく，概ね 7 日周期の成分である．違いは，徐々に自己相関係数が減衰し，概ね 2000 日後からは相関が認められなくなる．よって，2000 日以後は，季節要素の影響を極めて小さくする必要がある．以前の筆者らの研究 [10] では，1 日目から最後まで季節要素が同じように効果を及ぼすモデルであったため，学習が終わって予測を行う際に，季節要素が大きくなりすぎて予測に失敗している．

以上より，JDT の故障履歴はトレンド傾向を示し，2000 日目程度まで 7 日の周期を持つデータであることがわかったため，モデルにトレンド要素と，データの最初から 2000 日の期間のみ，7 日周期の季節要素を盛り込む．

これらのデータを用い，状態空間モデルとベイズ統計機械学習による故障発生件数の予測を試みた．藤原 [1] によれば，テストが中盤に差し掛かるか，テスト工程を 60 %程度消化するまでは，モデルのパラメータ推定は行わないほうが良い．そのため，筆者らの研究 [10] において 60%以上のデータを用いて解析したケースである，75%分のデータを用いて学習し，残りの区間における故障件数を予測するケースの解析を実施し，SRGM による予測結果と比較する．予測に用いたモデルは次節に示す．

4.3. 解析モデル

4.3.1 遅延 S 字型トレンドモデル

遅延 S 字型 SRGM を取り込んだ状態空間モデルを以下に示す．本モデルは，トレンド要素（$trend$）を，遅延 S 字型 SRGM の強度関数である式 (2) を変形することで作成し，7 日周期を表現する季節要素（$season$）と，コミット履歴に依存するコミット要素（$logcommit$）から対数故障強度（$loglambda$）が成り立つとした．

$$
\begin{aligned}
trend_{[t]} &= trend\_lin_{[t]} + \log(trend\_log_{[t]}) \\
trend\_lin_{[t]} &= trend\_lin_{[t-1]} - b \\
trend\_log_{[t]} &= trend\_log_{[t-1]} + a * b^2 \\
\sum_{l=0}^{6} season_{[t-l]} &= \mathrm{Norm}(0, \sigma_{season}) \\
logcommit_{[t]} &= logcommit_{[t-1]} + \mathrm{Norm}(0, \sigma_{commit}) \\
loglambda_{[t]} &= trend_{[t]} + season_{[t]} \\
&\quad + (c_1 * logcommit_{[t]} + d_1) \\
C_{[t]} &= \mathrm{Poisson}(\exp(logcommit_{[t]})) \\
Y_{[t]} &= \mathrm{Poisson}(\exp(loglambda_{[t]}))
\end{aligned}
\tag{11}
$$

式中の $C_{[t]}$ は，日付 t におけるコミット回数である．また，$\mathrm{Norm}(\mu,\ \sigma)$ は平均 μ，標準偏差 σ の正規分布，

Poisson(λ) は平均 λ のポアソン分布である．

季節要素は，任意の連続した 7 つの $season[t]$ を足すと，非常に小さな値 $\mathrm{Norm}(0, \sigma_{season})$ になるとした．これにより，7 日周期の変動を表現することができる．

コミット要素は，毎時のコミット履歴（$C_{[t]}$）から毎時のコミット発生回数の平均値を推定したものである．これを c_1 倍し，さらに d_1 を足すことで対数故障強度 $loglambda$ 算出に用いている．

コミット回数と故障発生件数は，ポアソン分布に従うとした．ポアソン分布は計数データを表現する標準的な確率分布であると同時に，NHPP モデルに用いられているためである．

対数故障強度 $loglambda$ は，トレンド要素，季節要素，コミット要素（定数倍・定数加算済）の線形和とし，対数故障強度の指数を取ったものをポアソン分布に適用することで，故障発生件数となるモデルとした．

4.3.2 対数ポアソン実行時間トレンドモデル

遅延 S 字型トレンドモデルの，トレンド要素の式を以下のように変えたものである．これは，対数ポアソン実行時間モデルの強度関数である式 (4) を変形したものである．式中の λ_0 と θ の意味は，同式と同じく，各々初期故障強度と故障 1 個あたりの故障強度減少率である．

$$trend_{[t]} = -\log(trend\_log_{[t]} + 1) + \log(\lambda_0) \quad (12)$$

$$trend\_log_{[t]} = trend\_log_{[t-1]} + \lambda_0\theta \quad (13)$$

4.4. 解析結果

前節にて示したモデルを，Google Colaboratory 上で，Pystan(ver.2.19.1.1) を用いて推定を行った．Pystan は MCMC 法に属する HMC 法と変分法に属する ADVI を使用できるが，計算時間の問題から ADVI を用いた．

推定した対数故障強度 $loglambda$ の中央値に指数関数を適用し，累積和を取ることで平均値関数 $H(t)$ とした．各 SRGM も，75%の故障発生件数データを学習データとして残りの故障発生件数を推定し，ベイズ統計機械学習の結果と比較した．SRGM は，学習データに対して最小二乗法でフィッティングし，予測に用いた．

4.4.1 遅延 S 字型トレンドモデル

結果を図 7 に示す．図中の黒色の実線は，累積故障発生件数実績値である．また，橙色の実線はベイズ統計機械学習による解析結果，2 本の点線は SRGM （遅延 S 字型 SRGM，対数ポアソン実行時間モデル）による解析結果である．なお，学習データと予測対象データの境界は，図中に赤色の点線で示した．また，図 8 は，対数領域における $loglambda$ の各要素への分解結果である．モデル内のパラメータの推定結果を，表 3 に示す．

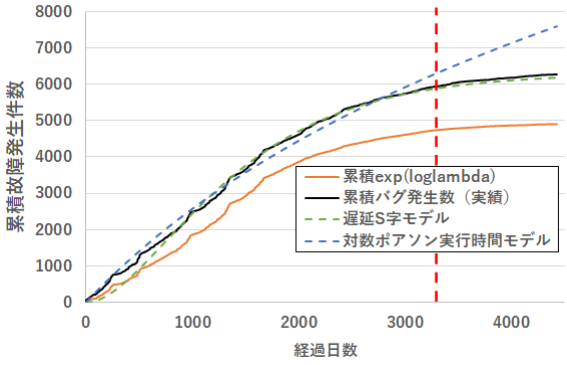

図 7. 故障発生件数の予測結果

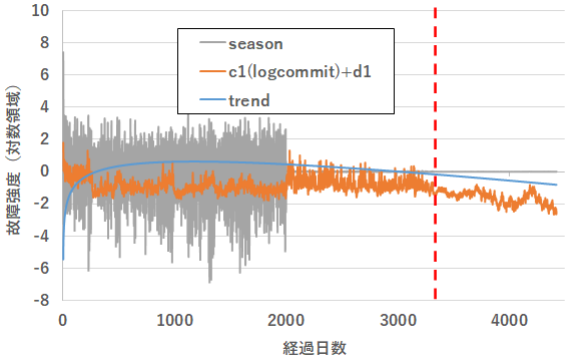

図 8. loglambda の各要素への分解結果

表 3. パラメータの推定結果

a	b	c_1	d_1
5955.49	0.000851	0.853769	-0.11752

本ケースにおける解析結果は，学習時点の間から実績の故障発生数より低めであり，SRGM，特に遅延 S 字型

SRGM が学習期間・予測期間ともに高い精度を示した．また，実績の総故障発生数は 6270 だが，予測総故障数である a の推定結果は 5955.49 と低めである．

4.4.2 対数ポアソン実行時間トレンドモデル

結果を図 9 と図 10 に示す．図中の線の意味は，遅延 S 字トレンドモデルと同じである．また，モデル内のパラメータの推定結果を，表 4 に示す．

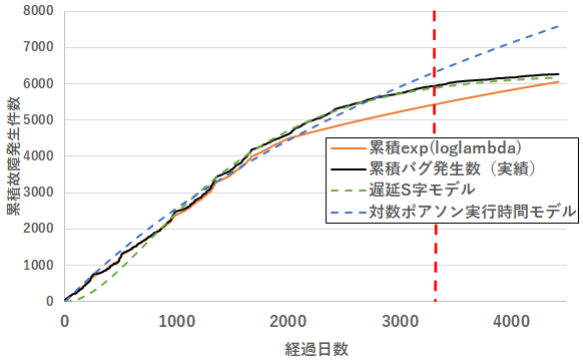

図 9. 故障発生件数の予測結果

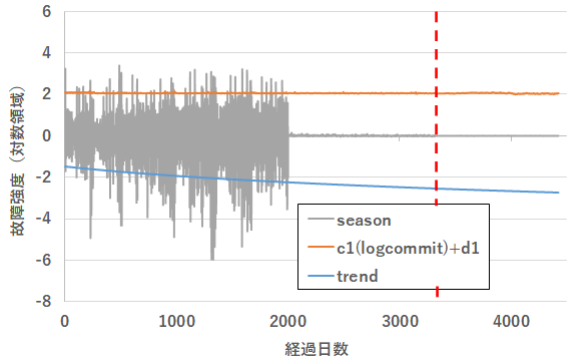

図 10. loglambda の各要素への分解結果

表 4. パラメータの推定結果

λ_0	θ	c_1	d_1
0.227249	0.002493	0.04024	2.09571

予測期間中において，対数ポアソン実行時間モデルよりも良好な予測精度を示しており，信頼性が安定しているという判断は正しく行えると考えられる．遅延 S 字型 SRGM よりも予測精度は下回ったが，500 日目までの複雑なバグの発生履歴には，SRGM2 種よりも高い追従性を示している．また，コミット要素に定数倍と定数加算を施した要素は，対数領域において，毎時ほぼ一定となった．

5. 考察と課題

提案手法である対数ポアソン実行時間トレンドモデルが，元の SRGM よりも学習期間中の一部を除き，高い予測と追従の精度を見せた．学習期間も，図 9 における 500 日目までの複雑な故障の発生履歴によく追従している．これは，7 日周期の季節要素をはじめとする，SRGM に無い要素の効果である．2000 日付近の，季節要素を操作した範囲で実績累積バグ発生件数と推定値に乖離が生じてしまったが，この箇所は季節要素の影響をゼロにする操作を開始した箇所であり，本操作が結果に悪影響を及ぼしている可能性がある．季節要素もトレンドのように減衰させるなどの工夫すれば精度の良い推定，さらには予測が可能になり得る．これは，バグレポートが作成される数の曜日ごとのムラの性質を把握できるようになることを意味し，信頼度成長曲線の横軸にテスト工程消化具合を使うのが難しく，日数を使う場合に OSS の信頼度を表現する大きな助けになると考えらえる．OSS の信頼度を的確に表現し成長を予測できるようになれば，OSS 採用を採用しようとしている組織にとって，採用可否や採用時にさらなる信頼性の作り込みを要するか否かの意思決定にも有用であると考えられる．そのためには，JDT 以外に様々な OSS，特に JDT と異なる故障の発生傾向を持つ OSS に対して本稿における提案手法を使用し，有効性を確かめていく必要がある．

反面，遅延 S 字型トレンドモデルは，元の SRGM よりも低い精度と追従の精度となってしまった．これは，遅延 S 字型 SRGM において，ソフトウェアが内包する予測故障数をパラメータ a として持っているため，SRGM 以外の要素も故障に関係すると考える本稿におけるアプローチは相性が悪いためであると考えた．a の推定結果と，総故障発生数のギャップもこのためであると考えられる．対して，対数ポアソン実行時間モデルは，故障強度が初期故障強度と時間だけに依存して減少するモデルであるため，他の要素を組み込むことができたと考えられる．この仮説を実証するならば，例えば，似た性質のワイブル過程モデル等の SRGM を用いてモデルを構築

して解析する必要があるだろう．

本稿におけるアプローチは，状態変数を用いてデータを要素に分解するものである．本アプローチは，経済・経営学 [16] 等，幅広い分野の時系列や空間データ解析に活用されている．よって，このようなデータと，適切な要素を具備したモデルを構築できれば，信頼性に留まらずソフトウェア工学の諸分野に適用できると考えられる．

図 10 にて，コミット要素を変換した要素が毎時ほぼ一定になり，トレンド要素が減少傾向になった．これは，故障強度が減衰するトレンド成分と，コミット履歴から推定された，毎時一定の成分に分解されたと考えられる．

三中が指摘するように，統計モデルによるデータの説明は，科学の方法論におけるアブダクションに従っている [5]．このアブダクションとは，複数の対立するモデルの中から最も良いモデルを選び出すこと（モデル選択）を指す [5]．本稿におけるモデルでも，SRGM の選択や，要素の取捨選択など，モデル選択の選択肢は多岐に渡る．その中で効率的に解析を行うための手法として，周辺化対数尤度や AIC を取得データと考えての直交表実験といったものが考えられる．直行表実験を使用するには様々な条件があるが，使用可能であればモデル選択の大きな助けになる．そのため，実験計画法の使用可否の検討や，モデル選択にも使える，実験計画法に相当する手法の構築が必要であろう．

参考文献

[1] 藤原隆次（著），木村光宏（編），信頼性技術叢書編集委員会（監）：ソフトウェアの信頼性，日科技連，2011

[2] ISO/IEC 25010 Software Product Quality：http://iso25000.com/index.php/en/iso-25000-standards/iso-25010（2021 年 5 月 21 日確認）

[3] 萩原淳一郎，瓜生真也，牧山幸史（著），石田基広（監）：基礎からわかる時系列分析 R で実践するカルマンフィルタ・MCMC・粒子フィルタ，技術評論社，2018

[4] 貝瀬徹:ワイブル分布に基づく階層的ベイズモデルの状態空間表現とフィルターによる推定，信頼性シンポジウム発表報文集，第 13 巻 (2000)

[5] 三中信宏：統計思考の世界 曼荼羅で読み解くデータ解析の基礎，技術評論社，2018

[6] J.D.Musa and K.Okumoto: A Logarithmic PoissonExecutionTime Model for SoftwareReliability Measurement, ICSE ’84: Proceedings of the 7th international conference on Software engineering, 1984, pp.230-238

[7] 日本規格協会：JIS Z 8115 ディペンダビリティ（総合信頼性）用語, 2019

[8] 野村佳秀，木村巧作，福寄雅洋，谷田英生：『オープンソースソフトウェア工学シリーズ』企業における OSS 活用の実際，コンピュータソフトウェア，Vol.33，No.3(2016)，pp.50-65.

[9] 尾崎俊治：非定常ポアソン過程モデル，情報処理，Vol.31，No.12(1990)，pp.1631-1640

[10] 杉山透，中谷多哉子：ベイズ統計機械学習を用いたオープンソースソフトウェア信頼度評価方の提案，信学技報 vol.120，No. 423（2021），pp.71-76

[11] 田村慶信，田中智朗，山田茂：オープンソースソフトウェアに対する SRGM に基づく予測精度の検証，数理解析研究所講究録，1636 巻（2009），pp.243-250

[12] 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, pp.19-22

[13] Shigeru Yamada, Mitsuru Ohba, Shunji Osaki：s-Shaped Software Reliability Growth Models and Their Applications, IEEE Trans. Reliability, Vol.R-33, No.4(1984), pp.289-292.

[14] 山田茂：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，オペレーションズ・リサーチ，Vol.61, No.10（2016），pp.666-667

[15] 山田茂，井上真二，田村慶信：ソフトウェア信頼性研究-モデリングアプローチ-，電子情報通信学会 基礎・境界ソサイエティ Fundamental Review，Vol.12, No.1(2018)，pp.38-50

[16] 山口類，土屋映子，樋口知之：状態空間モデルを用いた飲食店売上の要因分解，オペレーションズ・リサーチ，Vol.49，No.5（2004），pp.316-324