組込みソフトウェア製品開発のプロジェクト管理に対する遅延相関分析の適用に向けて

市井 誠[1,a)]　堀口 日向[2,b)]　柏 祐太郎[2,c)]　川上 真澄[1,d)]　伊原 彰紀[2,e)]　大平 雅雄[2,f)]

概要：多様化するソフトウェア開発プロジェクトを効果的に管理するためには，プロジェクト固有のソフトウェア進化に関する知識が必要となる．遅延相関分析は，時間的な遅延を挟んで現れるメトリクス間の相関関係を求めることができる，探索的な知識抽出アプローチであり，オープンソースソフトウェアのリポジトリに対する適用が報告されている．しかし，産業界の組込みソフトウェア開発では，製品ごとに異なるソフトウェア進化形態が形成されやすいため，遅延相関分析により抽出される膨大な相関関係を効率的に分析するための方法が必要となる．本稿では，産業界のソフトウェア開発のプロジェクト管理への適用に向けたアプローチを述べる．遅延相関分析は，得られた相関関係とメトリクスの時系列データとの対応を解釈することが難しいことを考慮し，相関関係を効率的に解釈するための可視化手法を提案する．また，実際の組込みソフトウェア製品の開発リポジトリを用いた適用実験を行い，プロジェクト管理に有用な相関関係が抽出出来ることを示した．

キーワード：遅延相関分析，ソフトウェアメトリクス，リポジトリマイニング

Towards Applications of Time-Delayed Correlation Analysis to Project Management of Embedded Software System Development Project

1. はじめに

大規模なソフトウェア開発プロジェクトのマネジメントには，データに基づき客観的に状況を把握し，意思決定するための技術が必要となる．ソフトウェアメトリクスは，その基礎的な道具であり，品質の見える化とマネジメント，プロセスの評価と改善など，様々なアプローチが発展してきた [1]．例えば，品質・コスト・納期（Quality/Cost/Delivery, QCD）に代表される，プロジェクト管理における関心事について，Goal-Question-Metric (GQM) パラダイム [2] を用いて具体的なデータやメトリクスへブレークダウンし，計測することで，その悪化の予兆を検知し，先手を取った対策をとれるようになる．

一方，産業界の組込みソフトウェア開発では，製品ごとに異なるアーキテクチャをもち，同じコードベースを長年開発し続ける傾向にあることから，独自のソフトウェア進化形態が形成されやすく，一般的な知見をそのまま適用することが難しい．プロジェクトマネジメントの観点・目的を，具体的なデータやメトリクスへと適切にブレークダウンするためには，一般的な知見だけでなく，プロジェクト固有のソフトウェア進化に関する知識が必要となる．

本稿では，産業界の組込みソフトウェア開発における予兆検知の実現に向けた，探索的な分析手法である遅延相関分析を用いたソフトウェア開発プロジェクトの分析アプローチについて述べる．遅延相関分析は，リポジトリマイニング [3] の一手法であり，ソフトウェアリポジトリから時系列データとして取得したメトリクスを分析し，時間的な遅延を挟んで現れるメトリクス間の相関を求める．Yamatani らは，オープンソースソフトウェア (Open Source Software, OSS) に対して遅延相関分析を適用し，リリースエンジニアリングなど OSS の開発活動をあらわす

[1] 株式会社日立製作所 研究開発グループ
社会システムイノベーションセンタ DX エンジニアリング研究部
[2] 和歌山大学 システム工学部
[a)] makoto.ichii.dn@hitachi.com
[b)] s216332@wakayama-u.ac.jp
[c)] kashiwa@ait.kyushu-u.ac.jp
[d)] masumi.kawakami.ch@hitachi.com
[e)] ihara@wakayama-u.ac.jp
[f)] masao@wakayama-u.ac.jp

相関関係を抽出した [4].

遅延相関分析手法は，メトリクスの時系列データに対する分析手法であり，OSS 開発を前提としたものではないが，産業界のソフトウェア開発プロジェクトに適用するには，解決すべき課題が存在する．製品リリースに向け複数の異なる工程を経てソフトウェア開発が進められるため，時期によって開発活動が変化し，ソフトウェア開発に関するメトリクスは非定常的に変動する．そのため，得られた相関関係には，それがよくあてはまる期間とそうでない期間が生じる．知識として活用可能な相関関係を得るためには，相関関係がよくあてはまる期間を把握し，意味のある関係かどうか精査しなければならない．しかしながら，遅延相関分析は対象となるメトリクス対に加算・差分・遅延の各係数を反映させた結果について相関を求めているため，元の時系列データのみでは，どのように相関が得られたのか理解困難である．実際の開発現場で活用するためには，妥当な労力で分析作業を行える必要がある．

そこで本稿では，遅延相関分析により得られた相関関係の効率的な解釈を支援するアプローチを提案する．提案手法では，メトリクスの時系列データが非定常的に変化することを考慮し，ヒューリスティクスによる相関関係の絞り込みおよび相関関係があてはまる期間の特定をおこなう．また，得られた相関関係の時系列データを，遅延相関の係数やあてはまり期間を反映させて可視化する．さらに，実際の組込みソフトウェア開発リポジトリの分析をおこない，遅延相関分析およびその可視化手法の有効性を評価する．

以下，2 節に遅延相関分析手法およびその課題について述べ，3 節にて提案手法について述べ，4 節にて組込みソフトウェア開発リポジトリへの適用およびその結果について述べる．5 節にて関連研究について述べ，最後に 6 節にてまとめと今後の課題について述べる．

2. ソフトウェア開発リポジトリに対する遅延相関分析

本節では，Yamatani らのアプローチ [4] の概要を述べるとともに，予兆検知を目標として産業界のソフトウェア開発リポジトリに適用する上での課題を述べる．

2.1 遅延相関分析の概要

遅延相関分析では，説明変数の値が累積し，一定期間の遅延をもって目的変数の値に変化を与えると想定する．その模式図を図 1 に示す．図 1 において，e_i, e_j をそれぞれ時刻 i, j における説明変数の値，o_n, o_m をそれぞれ時刻 n, m における目的変数の値とする．本稿では週を時刻の単位とする．説明変数 e_{ij} および目的変数 o_{nm} は下記の通り定義される．

$$e_{ij} = e_i + \cdots + e_j \tag{1}$$

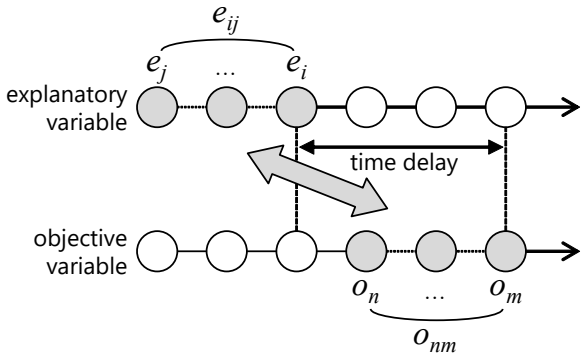

図 1 遅延相関分析の概念
Fig. 1 Concept of time-delayed correlation analysis

$$o_{nm} = o_m - o_n \tag{2}$$

説明変数の累積の期間を加算係数，目的変数が変化する期間を差分係数，説明変数の累積が目的変数に与える変化が完了するまでの期間を遅延係数と呼び，それぞれ，$i-j$, $m-n$, $m-i$ で表される．3 つの係数がすべて 0 のときは通常の相関係数である．

これら 3 つの係数の値域を $[0..M-1]$, メトリクスが N 種類存在するとしたとき, 遅延相関分析手法は $N(N-1)\times M^3$ 種類の相関係数を計算し，相関係数の絶対値の最も高くなる係数の組を，遅延相関として得る．

Yamatani らは，OSS 開発で用いられている Bugzilla および Git リポジトリから取得した 100 種類のメトリクスの時系列データに対し，外れ値除去および正規化の前処理を行った上で上記相関の計算をおこない，後処理として無相関検定を行った結果を精査し，リリースエンジニアリングなど OSS の開発活動を反映した相関関係を得た．

2.2 予兆検知への応用に向けて

遅延相関は，直感的には，（それ自身は直接的に悪いこととは限らない）何らかの状態が続いた後，一定期間の遅延をもって，何らかの注目すべき悪い事象が起こる，ということを表現できる．ソフトウェア開発においては，例えば，技術的負債の蓄積による開発工数の増大や，開発者の高負荷状態下での品質の悪化など，通常の相関分析では抽出しにくい関心事の予兆を得られると期待できる．

その一方で，実際に適用するには，解決すべき課題が存在する．まず，産業界のソフトウェア開発では，製品リリースに向け複数の異なる工程を経てソフトウェア開発が進められるため，時期によって開発活動が変化し，ソフトウェア開発に関するメトリクスは非定常的に変動する．これにより，得られた相関関係がよくあてはまる期間とそうでない期間が生じる．

例として，あるプロジェクトでの 2 つのメトリクスの時系列データを図 2 に示す．このうち，実線で示されるコミット数は前半と後半で値域が大きく異なるのがわかる．破線で示されるチケット対応時間中央値も，基本的には小

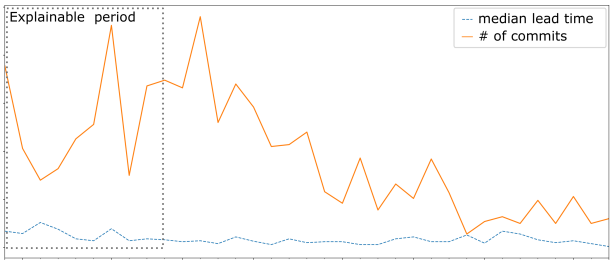

図 2 メトリクス値の非定常的な変化の例
Fig. 2 Example of unsteady change of metric values

さな値が続くが，期間開始付近，中央付近，および期間終了付近にて，他の期間と比較して大きな値をとる期間が存在する．このメトリクス対に対して週を単位として遅延相関を求めると，加算係数 1，差分係数 3，遅延係数 2，相関係数 0.31 の弱い相関関係が得られる．

分析者は，得られた相関関係を知識として活用可能なものとするため，背景にある事実を調査し，再現性のある事象かどうか精査を行う必要がある．しかし，メトリクスの非定常的な変化により，相関関係があてはまるかどうかは期間によって差があるため，あてはまりの悪い期間を調査してしまうと，見当違いの解釈をしてしまう可能性がある．図 2 の例では，点線で囲った期間があてはまりの最も良い期間であるが，これを時系列データから読み取ることは困難である．

このような変化点は，開発工程の変遷に基づくものであるが，その時期を機械的に得ることは難しい．まず，製品の開発現場によって開発プロセスが異なるため，どの工程がどのような影響をメトリクスに及ぼすか事前に把握しておく必要がある．さらに，製品の開発スケジュールは，市場の状況や顧客の要望，製品開発の進捗や品質状況に従って随時修正されるのに対して，その管理方法は開発現場ごとに大きく異なり，スプレッドシートなどを用い非形式的に管理されることも多いため，機械的に得ることが難しい．そのため，非定常的に変化するメトリクス値をそのまま入力できる分析手法が必要となる．

また，精査の際には，メトリクスの時系列データの値そのものではなく，説明変数は加算係数を，目的変数は差分係数を，それぞれ反映させた値について，遅延係数を考慮して事実との照合を行う必要がある．遅延相関分析では多数のメトリクス対に対する相関関係が得られるため，それら一つ一つについて，元の時系列データのみから精査を行うのは非現実的である．

そこで本稿では，遅延相関分析により得られた相関関係を効率的に解釈するため，あてはまりを理解可能な相関関係の可視化手法を提案する．相関関係にあるメトリクスについて，時系列データの対をプロットするとともに，遅延係数と差分係数を反映させた値を合わせてプロットする．また，相関が良くあてはまる期間を導出して出力する．さらに，いくつかのヒューリスティクスにより，信頼性の高い相関関係へと絞り込むことで，精査すべき相関関係を限定する．

3. 提案手法

本稿では遅延相関関係の抽出にあたり，Yamatani らの手法 [4] を改変して用いる．

3.1 遅延相関関係の抽出

3.1.1 前処理

時系列データに含まれる外れ値による解析への影響を防ぐため，箱ひげ図を用いて外れ値を除去する．ただし，ソフトウェア開発に関するメトリクス値は，対数正規分布やべき分布のような偏った分布になることを考慮し，通常，第一四分位点および第三四分位点を用いるところを，それぞれ 10 パーセンタイルおよび 90 パーセンタイルに置き換える．また，除去された値および元々の欠損をスプライン補間により補う．

また，下記の式にて時系列データを 0 から 1 の間に正規化する．

$$x_{i_{nor}} = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{3}$$

式 (3) において, x は入力となる時系列データ（メトリクス値），x_i および $x_{i_{nor}}$ は 時刻 i における値および正規化された値，$\max(x)$ と $\min(x)$ はそれぞれ x の最大値および最小値である．

3.1.2 遅延相関の算出

遅延相関を 2.1 節にて述べた方法にて算出する．相関係数の計算方法にはスピアマンの順位相関係数を用いる．

3.1.3 無相関検定

マンホイットニーの U 検定により，p 値が 0.05 を超える相関を除去する．

3.2 ヒューリスティクスによる遅延相関の絞り込み

大量に得られる相関関係を効率的に分析するため，以下に示す方法によって，興味深い結果へと絞り込む．

3.2.1 弱い相関の除外

より強い相関をもつ関係を優先的に分析するため，相関係数の絶対値が，ある閾値 α を下回るものを除外する．

3.2.2 遅延相関分析固有の結果への絞り込み

通常の相関分析で得られる関係，具体的には，加算係数=0 かつ遅延係数=0 となるもの，および，遅延相関係数が，通常の相関係数より低い値となるものを除外する．

3.2.3 相関係数の変化に基づく絞り込み

図 3 に示された 3 つのプロットは，それぞれ，あるメトリクス対の遅延相関関係に関し，相関係数の絶対値が最大となる加算/差分/遅延係数の組から，1 係数を変化させた

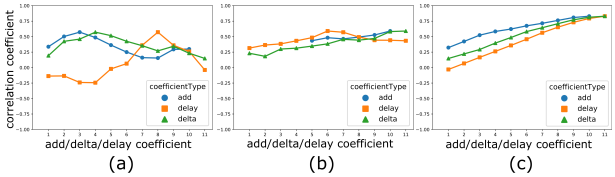

図 3 加算/差分/遅延係数を変化させたときの相関係数の変化
Fig. 3 Add/delta/delay coefficient and correlation efficient

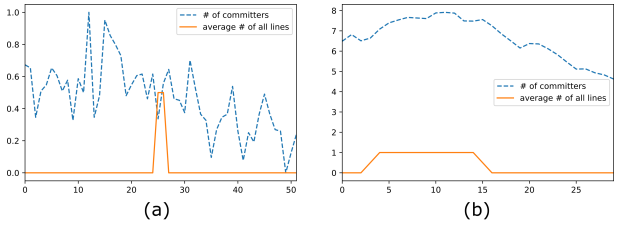

図 4 大きな加算/差分係数による平滑化
Fig. 4 Smoothing by large add/delta coefficient

ときの相関係数の変化を示したものである．(a) は意味のある遅延相関として期待する形状であり，相関係数が最大となる明らかなピークが存在する．一方，(b) は，明確なピークが存在せず，遅延の影響が小さい関係とみなせる．また，(c) のように単調増加となる場合は，平滑化により偶発的な事象が高い相関係数を得たものであると考えられる．単調減少の場合は，遅延の影響が相関を弱める方向に働くことから，通常の相関で得られる関係とみなせる．

本稿では，(a) のようなパターンを興味深い相関として抽出するため，このプロット，すなわち，2 係数を固定し，1 係数を $[1..M-1]$ の間で変化させたときの相関係数の変化がどのようなモデルの関数に従うかを求める．1 次関数，2 次関数および 3 次関数の 3 通りのモデルにそれぞれあてはめたとき，最も大きな対数尤度を示すモデルを採用する．加算/差分/遅延係数の全てについて，得られたモデルが 1 次関数である場合，(b) や (c) のような興味の無い関係であるとみなし，結果から除外する．

3.2.4 大きな係数の除外

大きな加算/差分係数は，偶発的な値を平滑化し，相関係数を不当に大きくすることがある．その例を図 4 に示す．(a) は 1 組の正規化済メトリクス値の時系列データであり，明らかに関連が見られない．しかし，加算，差分，遅延のそれぞれの係数を 10, 11, 11 とすると，図 (b) に示すように値が平滑化され，遅延相関係数が高い値（0.84）をとる．そのため，ある閾値 β を超える加算/差分係数をもつ関係を除外する．

また，実際のチケットの処理時間など開発プロセスの流れに照らし合わせたとき，過度に大きな遅延係数をもつ関係は，複雑な因果関係から生じたものである可能性がある．このような関係は解釈が困難であるため，遅延係数にも閾値 γ を設けることで，より有益な関係へと絞り込む．

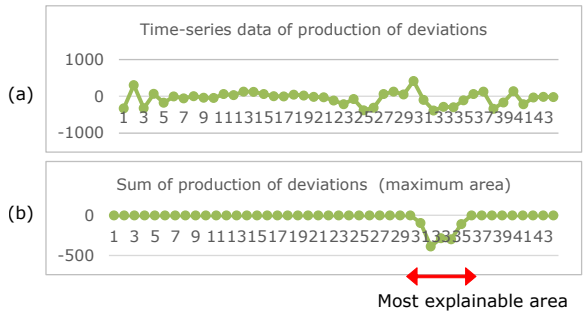

図 5 遅延相関のあてはまり期間の検出
Fig. 5 Derivation of Explainable Periods

3.3 相関のあてはまり期間の検出

2.2 節で述べた，メトリクス値の非定常的な変化により，得られた相関関係が良くあてはまる期間とそうでない期間が生じることを考慮して，本稿では，相関が良くあてはまる期間を「あてはまり期間」として特定・可視化する．分析者は，可視化されたあてはまり期間を参考に，得られた相関関係が偶発的なものでないか・再現性のある関係なのか判断する．あてはまり期間の特定にあたり，相関係数を算出する際に用いる，偏差の積の和を用いる．ここで相関係数 r は，下記の数式に示されるように，2 変数 x,y の共分散 s_{xy} を y の標準偏差 (s_x, s_y) で割った値であり，共分散，すなわち偏差の積の平均が 2 変数間の関係性を正負の相関として決定する．

$$r = \frac{s_{xy}}{s_x s_y} = \frac{\frac{1}{n}\sum_n^{i=1}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\frac{1}{n}\sum_n^{i=1}(x_i - \overline{x})^2}\sqrt{\frac{1}{n}\sum_n^{i=1}(y_i - \overline{y})^2}} \tag{4}$$

図 5 を用い，あてはまり期間を導出する方法を説明する．(a) は 1 対のメトリクスの偏差の積を時系列にプロットしたものである．これに対して，符号が一致する偏差の積の和を求め，その最大区間を示したものが (b) となる．この週番号 29—36 の期間が，相関係数を最もよく説明している期間として導出される．同様にして，2 番目以降の期間も導出することができる．後述する可視化のため，最大 5 件を抽出する．また，直感に一致した期間を提示するため，相関係数の正負に一致する箇所に絞り込む．

3.4 可視化

可視化イメージを図 6 に示す．横軸に時間をとるグラフであり，閲覧する遅延相関関係ごとに，(a) あてはまり期間，(b) メトリクス対の時系列データ，および (c) メトリクス対の時系列データに加算/差分/遅延の各係数を反映させた値の 3 種類のデータを，時点を揃え縦に並べて出力する．

(a) 3.3 節で述べた方法により求めた，あてはまり期間の上位 5 件を示す．遅延相関関係の説明変数となるメトリクスおよび目的変数となるメトリクスそれぞれについてバーで示す．また，遅延相関における加算係数・差分係数およ

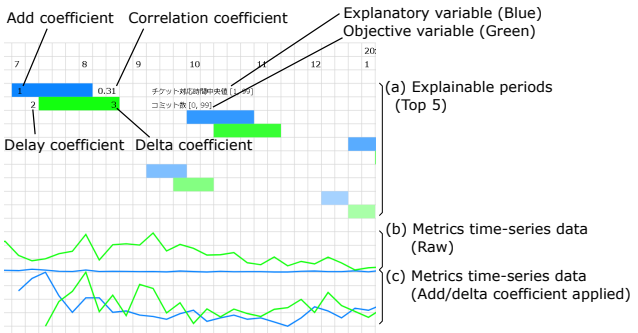

図 6 遅延相関の可視化
Fig. 6 Visualization of Time-Delayed Correlation

び相関係数も同じ区画に示す．(b) 遅延相関関係にあるメトリクス対の時系列データを折れ線グラフで示す．(c) 上記メトリクス対の時系列データについて，説明変数に加算係数を，目的変数に差分係数を反映した値を折れ線グラフで示す．

図では，チケット対応時間中央値と説明変数，コミット数を目的変数とする遅延相関が示されている．加算係数は 1，差分係数は 3，遅延係数は 2，相関係数は 0.31 であり，最上位のあてはまり区間は，ある年の 7 月第 1 週から 8 月第 1 週（目的変数は，遅延係数 2 を適用し，7 月第 3 週から 8 月第 3 週まで) であることが示されている．なおメトリクス名の後ろのカッコ書きは値域を示す．

4. 組込みソフトウェア開発リポジトリへの適用

提案手法の有用性の評価のため，実際の組込みソフトウェア製品の開発リポジトリへ適用する．まず，遅延相関分析で得られた知見が，将来のプロジェクトでも活用可能か評価するため，プロジェクトをまたがって存在する遅延相関関係がどの程度存在するか調査する．その上で，得られた遅延相関関係を精査することで，予兆検知に活用できる有用な遅延相関関係を得ることが出来るか評価する．

4.1 対象プロジェクト

ある大規模組込みソフトウェア製品の開発データ管理に用いているサービスである Redmine および Gitlab を解析し，メトリクスを抽出した．対象プロジェクトを表 1 に示す．A0 および B0 がメジャーリリースであり，A1 と A2，B1 と B2 は，それぞれ A0，B0 のマイナーリリースである．マイナーリリースを含め 3 ヶ月から 5 ヶ月に一度リリースされる開発ペースを考慮し，時刻の単位は週とする．

対象 Redmine プロジェクトでは，プロジェクト開始に先行したチケット作成や他プロジェクトからのチケット移動，プロジェクト終了時に残存したチケットの人手整理や Bot による自動処理が行われることから，プロジェクト期間を，チケットの作成・更新日付などから機械的に判定することが難しい．そのため，目視により開発活動が活発な期間を特定し，解析期間とした．具体的な方法は 4.3.1 節にて述べる．

なお，表 1 では，A0 の解析開始週を 1 週目としたときの週番号で解析期間を示している．

4.2 メトリクス

取得したメトリクスは，チケット，ジャーナル（チケット更新履歴)，コミットの 3 種類に大別される．コミットログおよびチケット，ジャーナルから取得したメトリクスの一覧を表 2 に示す．

ここで，チケットおよびジャーナルのメトリクスは，開発現場のワークフローを考慮して計測した．不具合修正にあたり，3 種類のチケットが用いられる．それぞれ，ここでは指摘管理チケット・不具合管理チケット・変更管理チケットとよぶ．ソフトウェアのテストにより不具合が発見されると，不具合管理チケットが発行され，原因分析が行われる．原因が特定され，修正方針が定まれば，変更管理チケットを作成してソースコードを修正する．開発工程が品質保証 (Quality Assuarance, QA) に移り，QA 検査にて問題が報告されると，不具合管理チケットに先立ち，対応状況の管理，および，トリアージ，すなわち，ソフトウェアの問題と，ソフトウェア外の問題，例えばハードウェアや文書の問題との切り分けのため，指摘管理チケットが作成される．ソフトウェアの問題であれば，テストで発見された不具合同様に不具合管理チケットおよび変更管理チケットが作成される．これらのチケット種別は，Redmine 上のトラッカーで区別される．このように役割の異なるチケットにてワークフローが組まれていることを考慮し，本適用例では，指摘管理チケット・不具合管理チケット・変更管理チケットそれぞれについて，作成数や更新数といったメトリクスを計測する．また，不具合管理チケットに関しては，QA 以降の検出か否かで重要度に大きな差があるため，開発者のテストで発見された不具合のチケットを不具合管理チケット（テスト)，QA 以降の指摘のチケット，すなわち指摘管理チケットから作成されるチケットを不具合管理チケット（QA）として，分けて計測する．また，チケット作成数については，総数の他，深刻度 (8 種) ごとに，個別

表 1 解析対象
Table 1 Target projects

プロジェクト名	解析期間（週番号）
A0	1 – 18
A1	26 – 42
A2	40 – 55
B0	57 – 91
B1	92 – 107
B2	109 – 134

に計測する．

チケットとプロジェクトは，Redmine のプロジェクトに基づき対応付ける．また，コミットとプロジェクトは，コミットメッセージ中に記載された Redmine チケット ID に基づき対応付ける．

メトリクス数は，コミット 10 種，チケット 16 種 (4×4)，ジャーナル 24 種 (6×4) である．

4.3 方法

4.3.1 遅延相関関係の算出

まず，プロジェクトごとに解析期間を設定する．プロジェクトごとに開発期間が異なることから，メトリクスを取得し，遅延相関を求める期間を，プロジェクトごとに定める．例えば，テスト開始前の不具合管理チケット作成数など，対応する開発活動がなくメトリクスの変化が無い期間が含まれると，外れ値除去や相関の算出が適切に機能しないことから，メトリクスに変化の見られる期間，具体的には全ての種類のチケットが 2 週間以上あくことなく起票される期間を，プロジェクトの期間とした（表 1)．なお，可視化の際は，解釈の参考とするため，これより広い期間，具体的にはいずれかのチケットが 2 週間以上あくことなく起票される期間のメトリクス値を出力する．

続いて，それぞれのプロジェクトに対して，3 節で説明した方法により，遅延相関関係の算出，絞り込みを実施する．ここで，加算・差分・遅延それぞれの係数の値域を [0..11] (約 3 ヶ月) とする．また，3.2 節 で絞り込みの閾値として，相関係数の絶対値の下限 α を 0.3，加算・差分係数の上限 β および 遅延係数の上限 γ を 3 週とする．

4.3.2 プロジェクトをまたがった遅延相関関係の調査

延相関分析で得られた知見がどの程度他のプロジェクトにて活用可能か評価するため，異なるプロジェクトにまたがって存在する遅延相関関係がどの程度存在するか，共通性を調査する．なお，このときの相関関係の同一性の判定にあたっては，加算・差分・遅延・相関の各係数の値は考慮せず，説明変数と目的変数のメトリクスの組のみを考慮する．

表 2 メトリクス
Table 2 Metrics

解析元	メトリクス
コミット	コミッター数，コミット数，変更ファイル数，変更行数，変更箇所数，関数数増減，変数数増減，サイクロマチック数増減，変更関数数，変更関数数（サイクロマチック数増加 5 以上）
チケット	作成数，作成者数，担当者数，対応時間中央値
ジャーナル	チケット { 情報変更回数，情報変更者数，担当変更回数，担当者数，担当割当者数，進捗更新回数 }

4.3.3 遅延相関関係の精査

得られた遅延相関関係を精査することで，予兆検知に活用できる有用な遅延相関関係を得ることが出来るか評価する．ここでは，複数プロジェクトに共通して見られる，QCD への影響，とくにここでは品質 (Q) への影響を示す関係を，有用な遅延相関関係として抽出することを目標とする．

具体的には，まず，以下に述べる方法により，遅延相関関係の候補を絞り込む．

複数プロジェクト共通 複数のプロジェクトに共通する相関は，偶発性が低く知識としての価値が高いと考えられる．具体的には，欠落もしくは相関係数の正負の逆転がたかだか 1 プロジェクトである相関関係に絞る

重要な目的変数への絞り込み QCD への影響の大きなメトリクスを目的変数とする遅延相関関係へと絞り込む．ここでは品質に関するものとして，指摘管理チケットに関するチケットメトリクス全 4 種およびジャーナルメトリクスの進捗更新回数を目的変数とする．

解釈困難な説明変数の除外 週当たりのチケット作成者数など，distinct (unique) カウントのメトリクスを説明変数とする遅延相関関係は，加算の解釈が難しいため，除外する

偶発的な結果の除外 可視化を用い，時系列データの目視から読み取れる関係と，得られた遅延係数および相関係数に明らかな齟齬が存在するものを除外する．例えば，相関係数が負の遅延相関について，異なる遅延係数での正の相関と捉えた方が適切な場合（もしくは，その反対）や，間欠的なメトリクス値の変化が均されて相関として検出されている場合がある．本稿では，相関係数の絶対値の大きなプロジェクト 2 つを選び，目視で妥当性を確認し，その両方に齟齬がある場合は，偶発的な結果であるとして，結果から除外する．

続いて，残った相関関係について，メトリクスの意味から，どういった開発状況を表したものか，解釈を検討する．

4.4 結果

4.4.1 プロジェクトをまたがった遅延相関関係

それぞれのプロジェクトに対して得られた相関関係を擬似的なベン図 [*1] で表したものを図 7 に示す．いずれかのプロジェクトに現れる相関関係として， 2,870 件が得られた．メジャーバージョンごとに見たとき，Ax 系 (A0, A1, A2) はいずれも約 1,900 件前後の相関関係が抽出され，それぞれの 30%程度が共通の相関関係として得られている．Bx 系 (B0, B1, B2) は，それぞれの検出数は 476 件，1592

[*1] 一部の組合せが示されないが，見やすさのためこの形態とした

表 3 得られた相関関係
Table 3 Extracted time-delayed correlation

#	説明変数	目的変数	A0	A1	A2	B0	B1	B2
1	不具合管理チケット（テスト）. 修正時間中央値	指摘管理チケット. 作成者数	0.512	0.531	0.322	0.300	-0.338	0.30
2	不具合管理チケット（テスト）. 修正時間中央値	指摘管理チケット. 担当者数	0.362	0.531	0.343	0.351	-0.344	0.30
3	変更管理チケット（テスト）. 修正時間中央値	指摘管理チケット. 担当者数	0.304	0.305	0.372	NA	0.343	0.475

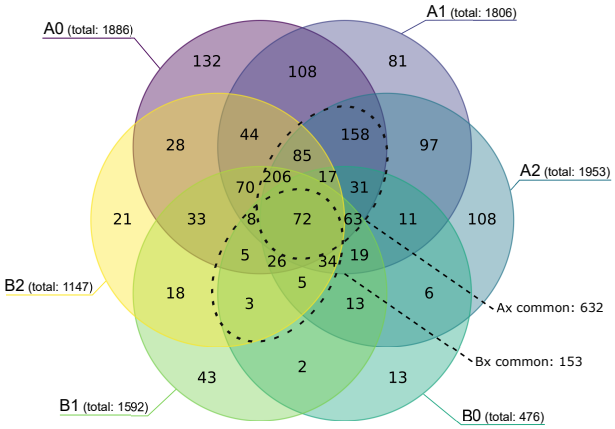

図 7 対象プロジェクト群から得られたの相関関係の共通性
Fig. 7 Commonarity of Time-delayed Correlations extracted from target projects

件，1147 件とばらつきがある．共通の相関関係 153 件は Ax 系と比較すると少ないが，B0 の件数が他に比べて少ないことの影響と考えられる．Bx 系共通の相関関係の B0 の相関関係に対する割合は約 32%であり，Ax 系と近い値となっている．また，全プロジェクトに共通する相関関係は 72 件得られた．個別のプロジェクト固有の相関関係は，13 件 (B0, 3%)〜132 件 (A0, 7%) 程度であり，いずれも少数である．これらのことから，過去のプロジェクトで得られた知見は，将来のプロジェクトでもある程度は活用できると考えられる．

4.4.2 遅延相関関係の精査

いずれかのプロジェクトに現れる相関関係 2,870 件のうち，欠落もしくは相関係数の正負逆転がたかだかひとつである相関関係は 108 件となり，さらに目的変数が指摘管理チケットに関するメトリクスに絞り，説明変数が distinct カウントであるメトリクスを除外すると，17 件となる．ここから，目視で偶発的な結果を除外すると，精査すべき相関として 3 件が残った．それらを 表 3 に示す．#1 の説明変数は，開発部署内で検出された不具合解決にかかる時間の中央値であり，この値の増大は，検出した不具合の多くを手間の掛かる難しいものが占めることを示す．また，目的変数は，QA にて指摘された不具合に関するものであるが，単なるチケット作成数ではなく，作成者数であることから，この値の増大は，複数の検査担当者にまたがり広く問題が生じている可能性を示す．#2 の目的変数も同様であり，複数の機能にまたがることを示唆する．また，変更管理チケットでは不具合管理チケットに基づく修正作業が行われるため，#3 の説明変数は #2 の説明変数と似た振る舞いをする．これらのことから，3 件は同一の状況を表していると考えられる．これらの解釈に基づき，テストで発見した不具合修正に普段より時間を要してしまった場合，テスト工程を無理に当初スケジュール通り終了させてしまうのではなく，QA 以降に影響の大きな不具合が生じる予兆とみなし，テスト工数の追加などプロジェクト管理での対策を検討すべき，といった議論が可能となる．

このうち，#1 について，相関係数の正負が異なる B1 を除く 5 プロジェクトそれぞれを可視化した結果を図 8 に示す．最も相関の高い A1 に対しては，変化の対応関係が明確に観測でき，B0 および B2 も弱い関係性を見て取れる．一方で，#1 のメトリクス対について，遅延を考慮しない通常の相関を，3 節で述べた前処理後の値を用いてスピアマンの順位相関係数により求めると，A0: -0.44, A1: -0.37, A2: 0.05, B0: 0.00, B1: -0.37, B2: -0.24 となり，それぞれ相関が見られないか，弱い負の相関，すなわち遅延相関分析と反対の傾向を示す．遅延相関分析の結果における，遅延・加算・差分の各係数をみたとき，最も相関の高い A1 では，説明変数の 3 週分の加算値と，目的変数の 3 週分の差分値が，2 週の遅延を以て相関する，という結果となっている．この結果は，あるマイルストーンに向け開発された機能が，開発部署でのテストを経て，QA にて検査を受ける，という開発の流れの時間感覚にあったものである．

4.5 考察

4.5.1 予兆検知への活用

本研究は，ソフトウェア開発リポジトリの遅延相関分析により産業界の組込みソフトウェア開発における予兆検知を目指したものであるが，そのゴールに対して有望な結果が得られた．関係性に遅延が存在することによって，遅延を考慮しない通常の相関分析では，相関無し，もしくは正負の逆転した相関関係として抽出してしまっていたものを，

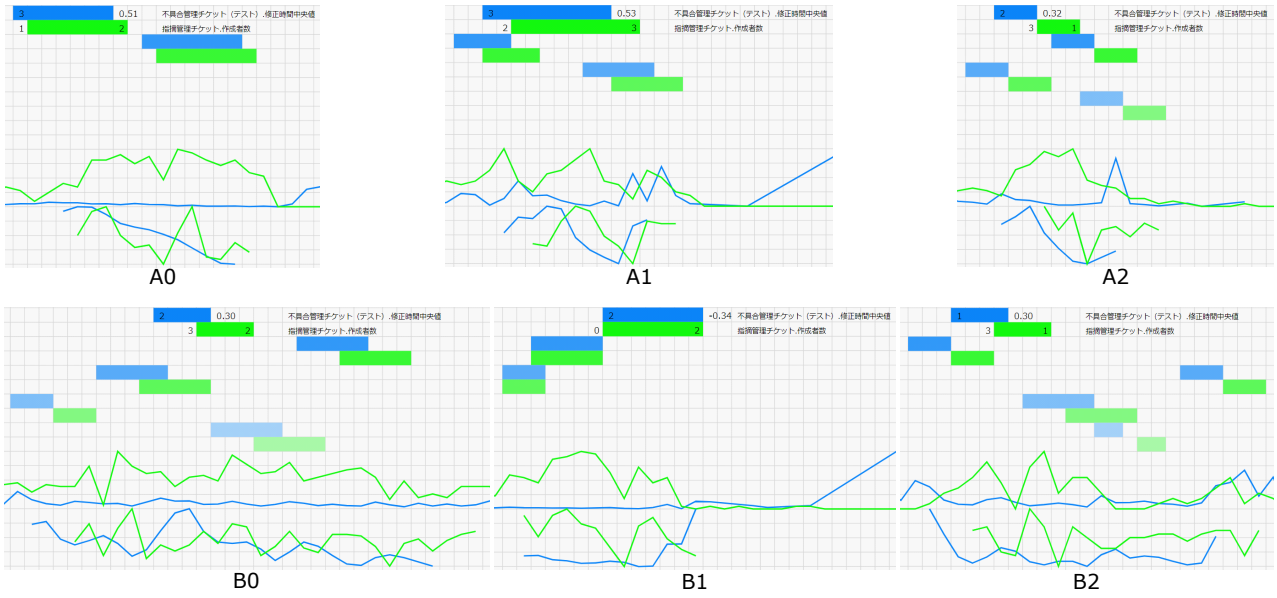

図 8 抽出された遅延相関（不具合管理チケット（テスト）. 修正時間中央値：指摘管理チケット. 作成者数）の可視化

Fig. 8 Visualization of extracted time-delayed correlation: Bug-ticket (testing).median lead time : Issue-ticket.# of reporters

遅延相関分析を用いることで，適切に抽出することができた．また，得られた相関関係の内容を検討したとき，QCD（品質・コスト・納期）の観点から，不具合管理チケットの対応時間中央値は，コスト（もしくは納期）に関する指標であり，指摘管理チケットの作成者数・担当者数は，品質に関する指標であるとみなせる．遅延相関分析によって，コスト（もしくは納期）に関する指標と，品質に関する指標との関係性が抽出できたのは興味深い結果である．品質は最も重視される指標であり，その悪化が検知されればコストをかけて対策がとられるが，その影響は，検知が早ければ早いほど小さく抑えることができる．従来より，時間の掛かる不具合の多発は，現場の経験則に基づき，品質悪化の予兆として報告・改善対策が行われてきたが，これを製品シリーズ共通の指標として継続的にモニタリングすることで，属人性やマネジメントの負荷を抑えながら，より早期に予兆を捉え，対策実施できるようになると考えられる．

本稿では，精査すべき相関係数を絞り込むために，プロジェクト間で共通する相関関係に注目したが，実際には，メジャーリリース時のみ（もしくはマイナーリリース時のみ）に現れる関係や，特定のメジャーバージョンで現れる関係などがあり得る．後述の課題に対処し，より的確な相関関係を機械的に抽出できれば，プロジェクトの特性を踏まえた，より踏み込んだ分析が可能になると考えられる．

本稿では，産業界の組込みソフトウェア開発の状況を想定してアプローチを組み立てたが，組込みシステム以外のソフトウェア開発においても，システム観点の検討や検証により開発に波がある場合など，メトリクスに非定常性を生むような開発活動の変化が存在する場合には適用可能であると考える．また，オープンソースソフトウェアについても，リリース品質を高めるため修正内容をコミュニティでコントロールしているなど，時期により開発活動に変化が現れる場合には，提案手法が有効に働くと考えられる．

4.5.2 分析作業にかかる労力

この結果を得るにあたっての分析作業は，十分に現実的な労力で行えた一方，今後の発展的な分析のための改善の余地も明らかになった．

まず，あてはまり期間および係数適用後のデータを添えた可視化により，精査の際に検討すべき期間を効率的に定めることができた．適用実験で得られた結果の解釈にあたっては，実際に開発現場にて注視すべき事象として認識されていたかどうかを確認するため，開発関連部署と製品開発責任者との週次定例会議の資料を調査した．正確な解釈を得るには，ある程度時間をかけて読み込む必要があるが，解析期間は A1 開始から B2 完了まで 134 週あるため，相関それぞれに対してその全てを読み込むことは現実的ではない．これに対し，提案手法の可視化を用いることで，まずは最も顕著な傾向があらわれたプロジェクト (表 3#1 では A1) の，最大のあてはまり期間に現れるピーク付近の数週程度を切り口として精査・解釈を進めることができ，労力と解釈の確からしさを両立させることができた．

一方で，より候補を広げた分析のためには，絞り込み過程での可視化結果の目視がボトルネックとなりうる．本稿での適用実験では，目視が必要となったのは 17 件のみであり大きな手間ではなかったが，精査対象をプロジェクト固有のものにまで広げるなど，規模を拡大する際には無視できない手間となる．本稿での適用実験にて，17 件中，正

しく関係性を表せていないと判断した 14 件について，その多くが，周期的な変化をするメトリクス対に対して，目視では，ある遅延係数にて正の相関関係が強いと判断されるが，解析結果では，異なる遅延係数にて負の相関が強いとして検出されていた．遅延相関を求めるとき，加算・遅延・差分の各係数の組み合わせに対して，相関係数の絶対値が最大となる組み合わせをとっているが，異なる極大値が適切であった可能性がある．これには，解析期間の設定において，全てのメトリクスに変化がある期間とした結果，解析期間が短くなり，時系列データの個数が少なくなったことも影響していると考えられる．データの個数を増やすには，期間を長く設定するか，時系列の単位を短くする必要があるが，期間を単純に長く設定すると，メトリクスの変化が無い期間の影響で適切な相関が得られなくなり，また，時系列の単位を 1 週間より短くとろうとすると，休日の影響が大きくなるため，現場のカレンダーを考慮した丁寧な前処理が必要になる．今後，極大値が正負にばらつく場合により適切な値を選択できるようなアルゴリズムの改良，また，十分なデータ数を確保するための，解析期間の設定手法の検討が必要となる．また，解析期間については，今回はプロジェクトごとに手作業で行ったため，これを自動化することはさらなる手間の削減にも繋がる．

5. 関連研究

リポジトリマイニングに基づくソフトウェア進化の分析に関して，様々な取組みが報告されている．Guo らは，Windows 開発におけるバグの再割当プロセスを調査し，再割当の理由やパターンを分析した [5]．Ihara らは，ボトルネックとなっている作業の発見などを目的とし，OSS のバグ修正プロセスを可視化している [6]．Xavier らは，Self-admitted technical debt (SATD) と呼ばれる，開発者自身が認識しコメント等に記載する種類の技術的負債について，バグ管理システムに基づき抽出する手法を提案している [7]. 市井らは，ソースコード中のアンチパターンを検出するにあたり，変更履歴に対するロジカルカップリング分析を用いている [8]．Xiao らは，アーキテクチャレベルの技術的負債を，変更履歴や依存関係を用いて抽出する手法を提案している [9]．Mo らは，変更履歴を用いたアーキテクチャ保守複雑性の指標を提案している [10].

6. まとめと今後の課題

本稿では，産業界の組込みソフトウェア開発における予兆検知の実現に向け，探索的な分析手法である遅延相関分析を用いたソフトウェア開発プロジェクトの分析アプローチについて述べた．遅延相関分析により得られた相関関係を効率的に解釈するための，相関関係の可視化手法を提案するとともに，実際の組込みソフトウェア開発リポジトリを用いた分析をおこない，遅延相関分析およびその可視化手法の有効性を評価した．結果として，予兆として活用可能な遅延相関関係を得ることができ，遅延相関分析が産業界のソフトウェア開発において興味深い結果を抽出可能であることが示された．今後の課題としては，得られた遅延相関のよりシステマティックな絞り込みと，他の製品ソフトウェア，特に異なるリポジトリ運用であるソフトウェアでの適用試行が挙げられる．

参考文献

[1] 阿萬裕久，野中　誠，水野　修：ソフトウェアメトリクスとデータ分析の基礎，コンピュータソフトウェア，Vol. 28, No. 3, pp. 3_12–3_28 (2011).
[2] 楠本真二，肥後芳樹：GQM パラダイムを用いたソフトウェアメトリクスの活用，コンピュータソフトウェア，Vol. 29, No. 3, pp. 3_29–3_38 (2012).
[3] 門田暁人，伊原彰紀，松本健一：ソフトウェアリポジトリマイニング，コンピュータソフトウェア， Vol. 30, No. 2, pp. 2_52–2_65 (2013).
[4] Yamatani, Y. and Ohira, M.: An Exploratory Analysis for Studying Software Evolution: Time-Delayed Correlation Analysis, *2014 6th International Workshop on Empirical Software Engineering in Practice*, pp. 13–18 (2014).
[5] Guo, P. J., Zimmermann, T., Nagappan, N. and Murphy, B.: "Not My Bug!" And Other Reasons for Software Bug Report Reassignments, *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, pp. 395–404 (2011).
[6] Ihara, A., Ohira, M. and Matsumoto, K.: An Analysis Method for Improving a Bug Modification Process in Open Source Software Development, *Proc. the Joint International and Annual ERCIM Workshops on Principles of Software Evolution and Software Evolution Workshops*, pp. 135–144 (2009).
[7] Xavier, L., Ferreira, F., Brito, R. and Valente, M. T.: Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems, *Proceedings of the 17th International Conference on Mining Software Repositories*, pp. 137–146 (2020).
[8] 市井　誠，川上真澄：リポジトリマイニングに基づくアンチパターン検出手法，情報処理学会論文誌，Vol. 64, No. 4, pp. 908–917 (2020).
[9] Xiao, L., Cai, Y., Kazman, R., Mo, R. and Feng, Q.: Identifying and Quantifying Architectural Debt, *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 488–498 (2016).
[10] Mo, R., Cai, Y., Kazman, R., Xiao, L. and Feng, Q.: Decoupling Level: A New Metric for Architectural Maintenance Complexity, *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 499–510 (2016).

[*1] Bugzilla は The Mozilla Foundation の米国およびその他の国における商標です．Git は，Software Freedom Conservancy, Inc. の米国およびその他の国における登録商標もしくは商標です．GitLab は，GitLab B.V. の米国およびその他の国における登録商標もしくは商標です．Redmine は，Jan Schulz-Hofen 及び Jean-Philippe Lang の EU およびその他の国における商標または登録商標です．Microsoft，Windows は，米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です．文中の社名，商品名等は各社の商標または登録商標である場合があります．なお，本文および図表中では，™，® マークは明記しておりません．