情報処理学会論文誌 Vol. 49 No. 8 2791–2801 (Aug. 2008)

企業横断的データからのプロジェクト改善案の相関ルールマイニングによる抽出

出張純也[†1] 水野修[†1] 菊野亨[†1]
菊地奈穂美[†2] 平山雅之[†2]

ソフトウェアの開発現場におけるプロジェクトの混乱を回避するために，プロジェクトのリスク要因を抽出する手法の開発が求められている．本研究では，ソフトウェアの品質に関する指標「不具合工数密度」に関して，その改善案を開発データから抽出する手法を提案する．具体的には，まずプロジェクトデータに相関ルールマイニングを適用することで「不具合工数密度」に影響を与える相関ルール群を抽出する．次に，抽出した相関ルール群から改善ルール（ルール中にほぼ同じメトリクスを含むが，「不具合工数密度」の評価値が異なるルール）群を特定し，その中から有用なプロジェクトの改善案を求める．本手法の提案および適用実験は産学連携研究として行った．適用実験として，日本国内で収集された企業横断的プロジェクトデータからのプロジェクト改善案の抽出を試みた．その結果，多くの改善案をほぼ自動的に抽出できることを確認すると同時に，その改善案が先行研究での指摘と基本的に一致することを示すことができた．

Mining Project Improvement Hints from Cross-company Data Using Association Rules

Junya Debari,[†1] Osamu Mizuno,[†1] Tohru Kikuno,[†1]
Nahomi Kikuchi[†2] and Masayuki Hirayama[†2]

In software project management, it is very important to identify risk factors which make software projects runaway. In this paper, we propose a method to extract improvement action items for runaway projects by applying association rule mining to the software project data. In the proposed method, we first mine association rules affecting the quality of software products from software project data. Then we group such rules that include common metrics with different values and based on the resultant rules we extract improvement action items. In order to evaluate the feasibility, we apply the proposed method to project data collected from plural companies in Japan. As the result, project improvement action items are semi-automatically extracted and their validity is confirmed by comparing the action items with the result of the previous research.

1. まえがき

1.1 研究の背景

ソフトウェアの開発現場ではプロジェクトの混乱状態の回避が急務となっている．しかし，開発プロジェクトの成功・失敗の要因は多岐にわたるため，その特定や特定した後の回避方法の検討は，経験をベースに行われることが多いが，回避方法選択の判断は定性的なことが多い．一方で，現場で観察収集される様々なデータを有効に活用してプロジェクトを制御する手法の確立が求められている．

実際の開発現場では，収集したデータはあってもプロジェクト制御に活用されていないことが多い．そもそも収集されたデータにはいわゆる雑音や欠損などもあり，その扱いが難しい．また，企業の生産活動の傍らでこうした問題に対処することは時間的にも大変難しい．そのため，より一般的に活用できるデータドリブン，または，ルール（経験）ベースのプロジェクト制御の考え方や方法の確立が必要とされている．

我々は開発現場から収集されたデータを利用したソフトウェア開発プロジェクトの最終状態予測に関する研究を行ってきた[6),11)]．しかし，これらの研究は，あるプロジェクトが混乱[7)]する可能性を確率などの形で示すにとどまっていた．そのため，動的な管理に適用することは難しかった．また，利用している混乱要因の数が少なく，かつ，固定的であったため，開発現場への大局的な指針を示すことは可能であるが，個々の問題に対処するには不十分であった．

そのため，プロジェクトの混乱要因を抽出し，それに基づいて開発現場へ何らかのフィードバックを行う手法の開発が求められている．我々はそのための手法として相関ルールマイニングの応用を検討している[13)]．しかし，相関ルールマイニングでは設定するパラメータによって大量のルールが抽出されたり，逆にほとんどルールが抽出されなかったりするなどの問題が存在していた．また，一般的に類似したルール（基本的にはルールの前提がほぼ同

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†2 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター（IPA/SEC）
Software Engineering Center, Information-technology Promotion Agency

じ要因を含んでいるルールのこと）が複数抽出されることが多いが，それらをうまくまとめなければ，現場へフィードバックすることができなかった．

本研究では，相関ルールマイニングを用いてリスク要因とプロジェクトの結果との関係をルールとして抽出し，抽出されたルールに基づいてリスク要因に対するプロジェクトの改善案などを提示する試みを行う．

具体的には，まず，プロジェクトから得られたデータに相関ルールマイニングを適用することで，プロジェクト悪化要因に関する多数の相関ルールを抽出する．次に，そのルール群から，プロジェクト改善への示唆を含むと思われるルール群を抽出する．この手法を利用することで，プロジェクト悪化の要因が，複数の要因を組み合わせた形で，分かりやすく見えてくると期待できる．また，この処理はデータを入力し，初期パラメータを設定すればほぼ自動的に行える．

1.2　産学連携の意義

上で述べたような手法の提案によってプロジェクトの改善案の抽出を試みようとした場合，産業界単独で，あるいは，大学単独でのアプローチは難しい．

ソフトウェア工学の問題としてとらえた場合，実学としての側面では実際の企業のデータと経験をベースにして得られた結果に対する検討を行うことが必須である．また，利用するデータについても実フィールドから得られたデータを利用しない限り，本手法の有効性は論じられない．

また，データマイニングや統計手法など，きちんとした科学的な分析をベースにしたアプローチが必要であり，そこから経験を知識に変換するプロセスが必要である．そのため，本研究の目的に対しては，産学の連携による解決を図る以外に有効な手段はないといえる．そうした背景をふまえて，本研究は研究機関と企業側の研究グループとの共同研究として実施された．産学の役割は次のようになる．

学術側の役割：雑音，欠損の多いデータの中から有効な知識をどのように抽出していくかなどの手法に関する提案を行う．この部分は統計的な手法の応用など様々な工夫が必要であり，産業界の知見だけでは解決できず，大学など研究機関の知見が必要と考えられる．

産業側の役割：意味のあるデータの選定と提供，雑音データなどの判断だけでなく，学術側から提供される手法の妥当性の判断も行う．これらは実際のプロジェクト経験に基づく判断が有効な部分であり，産業界の知見が必要である．

こうした体制の下，本手法の適用可能性を評価するための実験として，IPA/SEC から提供を受けた企業横断的データ[9] への手法の適用を試みた．実験の結果，企業横断的データからソフトウェアの品質に関わる興味深いルールを抽出できた．

本論文の構成は次のようになっている．2 章では，本研究の目的を述べる．3 章では，相関ルールマイニングの概要と研究で利用したデータセットについて述べ，既存研究との関連を説明する．4 章では，改善ルールの具体的定義や，導出方法について説明する．5 章では，実際のデータを使ったケーススタディとして提案手法を適用しプロジェクト改善案を抽出した結果について説明する．6 章では本研究の結論と今後の課題を述べる．

2. 研究の目的

本論文で提案する手法の目的は，(1) 大量のプロジェクトデータ中から監視対象に選んだプロジェクトの悪化要因に対応する説明変数を相関ルールの形で抽出し，(2) 得られた大量の相関ルールからプロジェクトの改善に役立つ知見を含むものを自動的に抽出する，という 2 点である．

目的 (1) に関する注意事項としては，相関ルールマイニングを用いる利点として，予想もしなかったような関係が相関ルールの形で得られること，欠損値を含むプロジェクトデータにも適用可能であること，などがあげられる．相関ルールマイニングでは次章で説明する支持度や信頼度といったパラメータを変化させることにより，抽出するルール数に一定の制限を掛けることができるので，処理しきれないほど大量のルールが抽出されるということは避けられる．また，マイニングの結果として得られたルールは，それぞれ何件のデータにおいて成り立っていたのかが明示的に分かるので，成立している事例数が多ければ傾向も強いと判断できるため，プロジェクトへの具体的なフィードバックがしやすいと考えられる．一方で，この段階で得られるルールには単独では現実的に意味をなさないものも多く含まれる．そのため，プロジェクトの改善に役立つようなルールを抽出することが必要となる．

目的 (2) に関しては，相関ルールの両立性に着目してルールの絞り込みを行う．基本的なアイディアは，両立しているルールであるがその結論が逆になっているもの（これを改善ルールと呼ぶ）を抽出することが，プロジェクトの改善につながると考えている．たとえば，

- レビューに費やす工数が『少ない』と最終的な品質が『低くなる』

というルールが存在したとする．ここで，改善ルールとは，

- レビューに費やす工数が『多い』と最終的な品質が『高くなる』
- レビューに費やす工数が『少ない』が単体テストを十分に行うと最終的な品質が『高くなる』

などを指すこととする．これらの改善ルールから，

- レビューに費やす工数を多くする
- 単体テストを十分に行う

などの対策を行うことで最終的な品質が上昇する，という改善案を求めることが可能である．両立しているルール，および改善ルールの具体的な定義は 4.1 節で述べる．

3. 準　　備

3.1 相関ルールマイニング

相関ルールマイニングとは，相関ルールと呼ばれる事象間の強い関係を知識として発見する分析手法である．$A \Rightarrow B$ という形で表される相関ルールは，A という事象が発生した場合に B という事象も発生するということを意味しており，A を前提，B を結論と呼ぶ．

この相関ルールの重要度を評価するパラメータとして,「支持度（support）」と「信頼度（confidence）」の 2 つがある．まず，支持度とはルールの出現頻度を表すもので，データ集合全体の中で A と B が同時に発生する確率（$p(A \cap B)$）である．また，信頼度とは事象 A が発生しているという条件下で事象 B が発生する確率（$p(B \mid A)$）を表す．つまり，この値が 1 に近づくほど，ルールの前提と結論の結び付きが強いことを意味する．実際のルール抽出では最低信頼度と最低支持度を設定し，その条件を満たすルールだけを抽出する．

相関ルールマイニングを用いることで，単一の要因が結果に与える影響だけでなく，要因間の関係も考慮に入れた「ルール」の抽出が可能となる．

3.2 企業横断的データ

本研究で利用した企業横断的データは IPA/SEC が収集したものである[9]．このデータは国内の企業 19 社から収集されたもので，汎用計算機上で動作するアプリケーションソフトウェアを開発するプロジェクトのものである．収集されたプロジェクトの形態は 9 割以上が受託開発である．それぞれのプロジェクトのデータには，規模・工数・工期・バグ数などの量的なプロジェクトデータ項目に加えて，プロジェクト特性を示す質的データが含まれている．収集されたメトリクスの詳細については文献 9) に詳しいため，ここでは省略するが，**表 1** に示す項目を含む多くのメトリクスが収集されている．プロジェクト数は 1,419 件であり，2005 年 6 月頃までに終了したものとなっている．

本研究で利用したメトリクスの一覧を表 1 に示す．これは，ルールマイニングを用いるにあたり，企業側の研究グループとの協議の結果，重要であると考えられるメトリクスのみを選定し，さらに，データ項目の欠損率が 90%以上のものを削除して分析用データセットを作成した結果である．これらのメトリクスは文献 9) のものそのままであるか，または後で説明するように，導出指標として加工して設定したものも若干数ある．なお，メトリクス名の後に (C)，(E)，(R) という記号を付けている．(C) はプロジェクトの実行時に，プロジェクト内で状況を改善するために対処策を行うなどして制御が可能であるメトリクスを表しており，これらは相関ルールの前提に利用するべきものである．(E) はプロジェクトまたは開発システムの環境条件に近いメトリクスを表しているため，これらのメトリクス

表 1　本研究で利用したメトリクス一覧
Table 1　Metrics used in this study.

新規顧客 (C)	新規業種・業務 (C)
新規協力会社_1 (C)	新技術利用 (C)
役割分担_責任所在 (C)	達成目標_優先度_明確度合 (C)
計画の評価（コスト）(C)	計画の評価（品質）(C)
計画の評価（工期）(C)	テスト体制 (C)
プロジェクト管理ツール_利用 (C)	構成管理ツール利用 (C)
設計支援ツール利用 (C)	ドキュメント作成ツール利用 (C)
デバッグ・テストツール利用 (C)	CASE ツール利用 (C)
コードジェネレータ利用 (C)	開発方法論利用 (C)
開発フレームワークの利用 (C)	要求仕様_明確度合 (C)
ユーザ担当者_要求仕様関与 (C)	ユーザ担当者_システム経験 (C)
ユーザとの役割分担・責任所在_明確度合 (C)	要求仕様_ユーザ承認有無 (C)
ユーザ担当者_設計内容理解度 (C)	設計_ユーザ承認有無 (C)
ユーザ担当者_受け入れ試験関与 (C)	PM スキル (C)
要員スキル_業務分野経験 (C)	要員スキル_言語・ツール利用経験 (C)
要員スキル_開発プラットフォーム使用経験 (C)	
開発プロジェクトの形態 (E)	利用形態 (E)
システム種別 (E)	業務パッケージ_利用有無 (E)
処理形態_1 (E)	アーキテクチャ_1 (E)
開発対象プラットフォーム_Linux (E)	開発対象プラットフォーム_Windows (E)
開発ライフサイクルモデル (E)	類似プロジェクトの参照の有無 (E)
要求レベル（信頼性）(E)	要求レベル（性能・効率性）(E)
要求レベル（セキュリティ）(E)	FP 実績値（調整前）(E)
月数（計画）(E)	月数（実績）(E)
プロジェクト開発工数計画値 (E)	平均要員数プロジェクト全体 (E)
ピーク要員数プロジェクト全体 (E)	品質保証体制 (E)
実効 SLOC 実績値 (E)	実績工数（開発 5 工程）(E)
月あたりの開発機能量 (E)	
顧客満足度_主観評価 (R)	プロジェクト成否_自己評価 (R)
実績の評価（コスト）(R)	実績の評価（品質）(R)
実績の評価（工期）(R)	発生不具合数 (R)
不具合工数密度 (FED) (R)	

は現状分析には有効であるものの，プロジェクトの改善には利用できない可能性がある．相関ルールの前提にこれらが存在するときはルールの解釈に注意を要する．最後に (R) はプロジェクトの結果で得られるメトリクスであり，相関ルールの結論になるべきものである．なお，(C) に相当するメトリクスは 31 個，(E) に相当するものは 23 個，(R) に相当するものは 7 個であった．

表 1 の開発対象プラットフォーム_Linux，および開発対象プラットフォーム_Windows は，企業側の研究グループとの協議の結果,「開発対象プラットフォーム[9)]」を Linux 系列と Windows 系列に分割したものである．また，月あたりの開発機能量を環境条件のメトリクスとして設定した．この指標は，FP を「FP 実測値_調整前[9)]」，S を月数（実績）とすると，次のようになる．

$$\text{月あたりの開発機能量} = \frac{FP}{S}$$

3.3 抽 出 対 象

研究機関と企業側の研究グループとの協議の結果，ソフトウェアの混乱に結びつく品質・コスト・期間の内，ソフトウェアの品質に関する指標の分析が急務であるとされた．そこで，本実験では対象とするメトリクスとして不具合の数を取り上げた．不具合とは，稼働後 6 カ月間に発見された問題であって，ソフトウェアの修正が必要となるものを指す．この不具合数を使って,「不具合工数密度（以下，FED と表記する）」を定義した．F を「発生不具合数[9)]」，E を「実績工数（開発 5 工程)[9)]」（開発全体の工数）とすると，次のようになる：

$$FED = \frac{F}{E}$$

この指標は，稼働後に発見された不具合数を，開発プロジェクトの大きさを表すために工数で正規化したものである．直感的にはこの指標の値が高いとプロジェクトの結果としてのシステムの品質が悪いということが推定できる．

なお，一般的な相関ルールマイニングでは連続値を持つ変数をルールの前提や結果の項目として扱うことはできないため，今回の実験では，元データのプロジェクトの FED の値に順位を付け，高い方から 1/3 を $High$，中央の 1/3 を $Middle$，低いほうの 1/3 を Low と分類した．

最終的に利用したデータの数は，この FED が計算できるもののみに限られた．そのため，利用したデータ数は 230 件となっている．

3.4 関 連 研 究

相関ルールマイニングは様々な分野で有益な情報を抽出するために用いられている． ソフトウェア工学の分野における利用例として，たとえば，Michail はアプリケーション内でライブラリが再利用されるパターンを相関ルールマイニングを用いて発見し，そのパターンをクラスライブラリの構築に反映させる試みを行っている[2)]．また，Zimmermann らは，バージョン管理システムに記録されている変更履歴データに相関ルールマイニングを適用することで，同時に変更されやすいファイルやモジュールの組合せを特定する試みを行っている[8)]．Morisaki らは相関ルールマイニングの拡張として，結論に連続値の範囲を利用できる手法を提案している[4)]．この手法は従来の名義的尺度を用いたルールマイニングでは検出できなかったルールを抽出できるという点は画期的であるが，ルールの現場へのフィードバックについては述べられていなかった．また，相関ルールのクラスタリングを行う手法としては Gupta らが距離を使ったクラスタリングを行った事例がある[1)]．大量に生成されるルールをまとめる手法としてクラスタリングは有効であるが，抽出したルールの正確さが失われるなどの問題がある．

企業横断的データを利用してプロジェクトに有益な情報を引き出そうとする試みもなされている[5),10),12)]．大杉らは文献 12) において協調フィルタリングを利用して企業横断的データからプロジェクトのコスト予測を行う手法を提案している．また，同様に文献 5) では Case Based Reasoning を用いてコスト予測を行っている．どちらの研究も IPA/SEC との共同研究であり，企業横断的データを用いた産学連携の代表的な事例である．本研究は品質を対象としてデータの分析を行っており，文献 5)，12) とは目的が異なっている．また，古山らは文献 10) において，企業横断的データに対するクロス分析を実施し，収集されたメトリクスからプロジェクトの成否に影響を与える要因の特定を行っている．本研究の立場は文献 10) に最も近いため，5.4 節では得られた結果の比較を行う．

4. プロジェクト改善案抽出手法

4.1 プロジェクト改善案

ここでは本研究でのアイディアであるプロジェクト改善案について述べる．そのために，まず「両立するルール」を次のように定める．

あるルール r が与えられたとき，r の前提が含むすべてのメトリクスの集合を M_r^A，結論が含むすべてのメトリクスの集合を M_r^C とする．同様に別のルール s の前提のメトリクスの集合を M_s^A，結論のメトリクスの集合を M_s^C とする．このとき $M_s^A \supset M_r^A$ かつ $M_s^C = M_r^C$

が成立するならルール s をルール r とメトリクスの集合に関して両立するルールと呼ぶ．

例 1　次のルールを考える．

- レビュー工数 $= Low \Rightarrow FED = High$

このルールと両立するルールの例は以下のようになる．

- レビュー工数 $= High \Rightarrow FED = Low$
- レビュー工数 $= Low \wedge$ テスト工数 $= High \Rightarrow FED = Low$
- レビュー工数 $= High \wedge$ テスト工数 $= Low \Rightarrow FED = High$

また，次のルールは前提のメトリクスが違っているため，両立するルールとはならない．

- テスト工数 $= High \Rightarrow FED = Low$

ここでは，次のようにして改善案を導出することを試みる．上で定めたルール r と両立するルール s に注目する．ルール r の結論のメトリクス FED の値は $High$ である．このときルール s の結論でメトリクス FED の値がルール r とは反対に Low あるいは $Middle$ となっているなら，このルール s を改善ルールと呼び，r に対する改善ルールの集合を R_r^* と表す．

例 2　例 1 と同じルールを考える．

- レビュー工数 $= Low \Rightarrow FED = High$

改善ルールとは，両立するルールのうち，FED が $High$ ではない（つまり Low か $Middle$ である）ものとなる．したがって，例にあげていた 3 つの両立するルールのうちで次の 2 つだけが改善ルールとなる．

- レビュー工数 $= High \Rightarrow FED = Low$
- レビュー工数 $= Low \wedge$ テスト工数 $= High \Rightarrow FED = Low$

こうして得られた R_r^* からプロジェクトの改善案を示唆するメトリクスの集合を次のように定める．R_r^* 中の各ルール t に対し，前提が含むすべてのメトリクスの集合 M_t^A を求める．最終的に求まるメトリクスの集合 $M_r^* = \bigcup_{t \in R_r^*} M_t^A$ を参考に，プロジェクト改善案を策定する．この部分は自動処理では行わず，経験のある管理者，開発者，研究者が意味のある改善案を策定する．

プロジェクトの改善案の考え方について説明する．両立したルールが存在するということは，開発の状況が似ていることを表している．この両立した状況の中で結論が逆になっているということは，前提にあげられているメトリクスの変化や，新たに追加されたメトリクスによってプロジェクトの状態が逆転するという可能性を示している．こうした事例をルールに含まれるメトリクスの集合 M_r^* という形で収集することは，プロジェクトの状況を変化させうる要因を収集することにほかならない．そのため，本研究ではこれらをプロジェクトの改善案と位置づける．

4.2　プロジェクト改善案抽出の手順

図 1 にプロジェクト改善案抽出の手順と，産学での分担を表す．

4.2.1　Step 1：データ提供と必要なメトリクスの検討

この Step は，開発現場からのデータ収集および必要なメトリクスの検討を行うため，企業側の研究グループが担当する．開発現場から提供されたデータはそのメトリクスの種類も多岐にわたるため，必要なメトリクスを適宜選択することが求められる．この Step ではこうしたデータの前処理を行う．

Step 1 のポイントは次のとおりである．

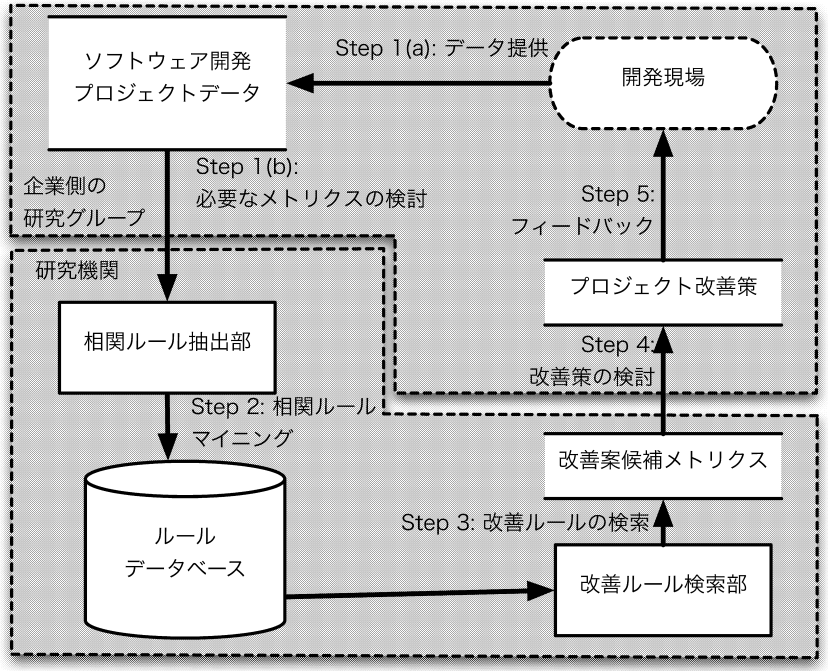

図 1　提案手法の概要
Fig. 1　Outline of proposed approach.

- 改善したいゴールを定め，ゴールを表すメトリクスの候補をいくつかあげる（5 章に示すケーススタディでは，品質の問題を指定した）．
- 変数となるメトリクスを選定し，収集データを精査する．
- メトリクスを，3.2 節で説明したように (C)，(E)，(R) の種類に分けておく．次の Step 2 以降において，(C) および (E) のメトリクスはルールの前提として使用され，(R) のメトリクスはルールの結論として使用されることになる．
- 収集データから切り出した分析するデータセットを研究機関とともにレビューして確定する．

4.2.2　Step 2：相関ルールマイニング

この Step は研究機関が担当する．この Step の主な流れは次のとおりである．

(1)　対象とするデータに対して，ルールの結論となる変数を FED の 1 つだけに限る．
(2)　最低支持度，最低信頼度を定める．
(3)　データに対して相関ルールマイニングを実施し，得られたルールをすべてデータベースへと格納する．

ここまでで，大量の相関ルールがデータベースに格納されることになる．次の手順ではその中から必要なルールを抽出する．

4.2.3　Step 3：改善ルール検索

この Step も本研究では研究機関が担当する．この Step の流れは次のとおりである．

(1)　ルールの結論が $FED = High$ となっているルール r に対し，次の条件を満たすルール r' を検索する．r の前提に存在するメトリクスをすべて含んでいて，r' の結論がメトリクス FED だけを含んでいる．そのような r' は複数存在するので，その集合を R'_r で表す．
(2)　$r' \in R'_r$ の結論が $FED = Low$ あるいは $FED = Middle$ になっていれば，r' を r^I（改善ルール）と表記して，それらの集合を R^*_r と表す．
(3)　最終的には R^*_r に残っているルールのメトリクスに基づいて改善案候補を作成する．

4.2.4　Step 4：プロジェクト改善案の検討

R^*_r から r に対する改善案候補が得られるので，その妥当性の検討を行う．メトリクスの持つ意味やその使われる状況なども考慮した検討を行うため，この作業は開発データやその背景に精通した者が実施することが望まれる．そのため，本研究では企業側の研究グループが担当する．

Step 4 の改善案は次の点に着目して検討する．

- 抽出された改善案候補のうち，ルールの前提では，3.2 節で述べた (C) のタイプのメトリクスは，プロジェクトで制御可能であることから，(C) タイプを優先的に改善策として採用を考慮する．
- (E) タイプのメトリクスを持つルールは，プロジェクトの制約条件が異なる可能性が高いため，改善案の検討候補には入れない．
- 改善案として見つけたルールについて，支持度と信頼度を参照し，成立している件数の多いもののほうを優先し，実際の改善をプロジェクトで実行する案の候補に入れる．

4.2.5　Step 5：フィードバック

Step 4 で得られた結果は最終的には開発現場へのフィードバックとしてプロジェクトの改善に利用されることになる．この Step は企業側が担当する．

5. ケーススタディ

ここでは，実際に IPA/SEC データ[9] について提案手法を適用し，プロジェクト改善案を抽出した結果について述べる．

5.1　適 用 結 果

本ケーススタディでは，IPA/SEC データ[9] に対し，3.2 節と 3.3 節で述べた準備を行い，表 1 のメトリクスとして整えたデータセットについて提案法を実行した．

まず，Step 2 では最低支持度 = 0.015，最低信頼度 = 0.9 としてルールマイニングを行ったところ，8,235 個のルールが求まった．次に，結論が $FED = High$ となるルール r に限定して Step 3 を実行し，この 8,235 ルールの中から改善ルールを抽出した．その結果，82 個のルールのそれぞれについて改善ルールの集合 R^*_r を抽出できた．なお，改善ルールの総数は 530 個となっていた．Step 2 と Step 3 に要した時間は 80 分であった．なお，実行環境の CPU は Intel Core 2 Duo 2.33 GHz，メモリは 3 GB である．

5.2　詳細な分析

改善ルールの集合 R^*_r の総数が 82 個なので，原理的には少なくとも 82 個の改善案が得られる．今回は慎重を期して各 R^*_r から 1 つまたは少数の改善案を取り出すことにした．

以下では，抽出された 82 個の改善案の中で特徴的な 3 つについて，詳細な分析を試みる．

5.2.1　プロジェクト改善案 1

次に示すルール（ここではルール r_1 と書く）がソフトウェアの品質を悪化させる要因として抽出された．このルールの支持度は 0.017，信頼度は 1.0 であった．

新規業種・業務 = 新規業種・業務
∧　計画の評価（品質）= 計画なし
⇒　$FED = High$

このルール r_1 は，プロジェクトが新規の業種・業務であり，品質に関して計画をしていないならば不具合密度が高くなることを示唆している．

一方，ルール r_1 に対して次に示す改善ルール r_1^I が求まっていた．このルール r_1^I の支持度は 0.017，信頼度は 1.0 であった．

新規業種・業務 = 新規業種・業務
∧　計画の評価（品質）= 品質目標が明確で実行可能性を検討済み
∧　テスト体制 = スキルと員数ともに十分
⇒　$FED = Low$

このルール r_1^I は次のことを示唆している．プロジェクトが新規の業種・業務であり，テスト体制が開発者のスキルと人数ともに十分で，品質目標が明確で実行可能性を検討済みである場合に不具合工数密度が下がる．

これらのルールから次に記すプロジェクト改善案が得られた．

- プロジェクトが新規の業種・業務である場合，品質目標を明確にし実行可能性を検討して，テストでは開発者のスキルと人数ともに十分な体制を構築することで，不具合工数密度が低くなる．

5.2.2　プロジェクト改善案 2

次に示すルール（ここではルール r_2 と書く）がソフトウェアの品質を悪化させる別の要因として抽出された．このルール r_2 の支持度は 0.017，信頼度は 1.0 であった．

開発プロジェクト形態 = 受託開発
∧　アーキテクチャ = 2 階層クライアント/サーバ
∧　ドキュメント作成ツール利用 = あり
⇒　$FED = High$

このルール r_2 は次のことを示唆している．受託開発のプロジェクトで，アーキテクチャが 2 階層クライアント/サーバであるソフトウェアを開発していて，かつ，ドキュメント作成ツールを利用している場合，不具合工数密度が高くなる．

一方，ルール r_2 に対して次に示す改善ルール r_2^I が求まっていた．このルール r_2^I の支持度は 0.017，信頼度は 1.0 であった．

開発プロジェクト形態 = 受託開発
∧　アーキテクチャ = 2 階層クライアント/サーバ
∧　ドキュメント作成ツール利用 = なし
∧　デバッグ・テストツール利用 = あり
⇒　$FED = Low$

このルール r_2^I は，ルール r_2 と同様に，受託開発のプロジェクトでアーキテクチャが 2 階層クライアント/サーバであるソフトウェアを開発している状況を示している．しかし，ルール r_2 とは違い，ドキュメント作成ツールを利用しないで，デバッグ・テストツールを利用している場合は不具合工数密度が低くなることを意味している．

これらのルールから次に記す改善案が得られた．

- 受託開発である 2 階層クライアント/サーバシステムの開発においては，ドキュメント作成ツールを利用しない場合に，デバッグ・テストツールを利用することで，不具合工数密度が低くなる．

5.2.3　プロジェクト改善案 3

次に示すルール（ここではルール r_3 と書く）がソフトウェアの品質を悪化させる別の要因として抽出された．このルール r_3 の支持度は 0.017，信頼度は 1.0 であった．

役割分担_責任所在 = おおむね明確
∧　構成管理ツール利用 = なし
⇒　$FED = High$

このルール r_3 は，役割分担責任所在に曖昧な部分があり，構成管理ツールの利用がない場合，不具合工数密度が高くなるケースを示している．一方，ルール r_3 に対して，次に示す 4 つの改善ルール $r_3^{I_1}$，$r_3^{I_2}$，$r_3^{I_3}$，$r_3^{I_4}$ が求まっていた．これらのルールの支持度はいずれも 0.017，信頼度は 1.0 であった．

- ルール $r_3^{I_1}$
  役割分担_責任所在 = 非常に明確
  ∧　構成管理ツール利用 = あり
  ∧　要求仕様_明確度合 = 明確
  ⇒　$FED = Low$
- ルール $r_3^{I_2}$

役割分担_責任所在 = 非常に明確
∧　構成管理ツール利用 = あり
∧　計画の評価（品質）= 品質目標が明確で実行可能性を検討済み
⇒　$FED = Low$

- ルール $r_3^{I_3}$
  役割分担_責任所在 = 非常に明確
∧　構成管理ツール利用 = あり
∧　計画の評価（コスト）= コスト算定の根拠が明確で実行可能性を検討済み
⇒　$FED = Low$

- ルール $r_3^{I_4}$
  役割分担_責任所在 = 非常に明確
∧　構成管理ツール利用 = あり
∧　計画の評価（工期）= 工期計画の根拠が明確で実行可能性を検討済み
⇒　$FED = Low$

これらのルールから次に記す改善案が得られた．

- プロジェクトでは役割分担と責任所在を非常に明確にし，構成管理ツールを利用する．さらに，要求仕様を明確にすること，計画時に品質目標を明確にすること，また，コストおよび工期の根拠を明確にして実行可能性を検討することが，結果として良い品質につながる．

5.3　得られたプロジェクト改善案

提案手法により，上で説明した 3 つの改善案を含む 82 個の改善ルールの集合 R_r^* が抽出された．抽出されたルールの集合から得られた改善案の候補を，企業側の研究グループが有用であるかを検討し，次に示す 10 個の改善案が有用であると判断された．

(1)　言語，ツール利用に関するトレーニングをプロジェクト初期に実施する，または，メンバに経験者を多少追加したり，コーチングを行ったりすることなどで，開発言語やツールを使いこなせるレベルに早急に上げていくことが必要である．
(2)　ドキュメント作成ツールを利用しない場合に，構成管理ツールを利用する．
(3)　ドキュメント作成ツールを利用しない場合に，デバッグ・テストツールを利用する．
(4)　プロジェクトマネージャのスキルが高い人を配置する．または，スキル不足の PM の場合は組織として PM サポート機能で支援する．
(5)　プロジェクト内での責任所在を明確にする．
(6)　計画時に，品質目標を明確にし，実行可能性を検討しておく．
(7)　ドキュメント作成ツールを利用しない場合に，プロジェクト管理ツールを利用する．
(8)　要求仕様を明確にする．
(9)　スキルのあるテスト要員を，十分な人数配置する．
(10)　計画時にコストの根拠と計画工数の妥当性を確認する．

82 個の改善ルールの集合から改善案を求める別の方法として，ルール r の支持度が高いものを優先的に抽出するという方法での分析も行った．その結果，抽出されたルールには，ルールの前提にほぼ同じ要因が含まれているものが多かった．したがって，さらに集約が必要であるということが判明したため，この方法を本研究では採用しなかった．

本ケーススタディで利用した IPA/SEC 収集のデータに対してクロス集計を中心に生産性の要求分析を行っている文献 10) では,「プロジェクト成功への指針」がまとめられている．その中で，品質に関しては 7 つの要因があげられている．その内容については，本ケーススタディで求めた改善案との比較を 5.4 節において行う．

5.4　関連研究との比較と考察

ケーススタディで利用した IPA/SEC 収集による企業横断的データについての分析を行った研究は 3.4 節でも述べたようにこれまでにも多くなされている．今回は，その中でもプロジェクトの結果に対する要因分析を行った研究[10] との比較を行う．

図 2 に文献 10) において発見された品質低下の要因と本研究で発見されたプロジェクト改善案の関連を示す．詳しく説明すると，図 2 の左側には文献 10) の 2,617 ページの 5.4 節の「プロジェクト成功への指針」の品質についてまとめた 7 つの要因を並べている．一方，右側には 5.3 節で求めた 10 個のプロジェクト改善案を並べている．そして，互いに関連する指摘事項どうしを線で結んでいる．

図 2 から分かるように，本研究で得られたプロジェクト改善案は，従来の研究[10] で示されていた品質確保のための 7 つの要因すべてに対応する．

互いに関連のある指摘事項のうち，ここでは 2 つの事項について詳細に説明する．古山らは文献 10) において品質低下の原因として以下の要因をあげている．

> 達成目標_優先度_明確度合いの高いプロジェクトおよび PM スキルの高いプロジェクトは，いずれも品質低下を起こしにくい傾向が見られる（いずれも 10%有意）

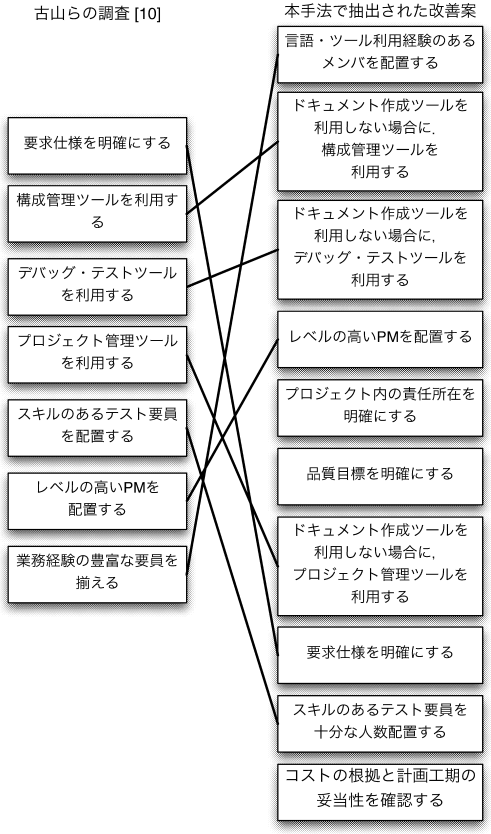

図 2　古山らの調査と本研究の比較
Fig. 2　Comparison between Furuyama's analysis and ours.

この要因に対応して，本研究では 5.3 節の (4) の改善案が抽出されている．
また，次のような要因も文献 10) では指摘されている．

> 品質低下の防止に有効と考えられるものは「ツール利用（プロジェクト管理ツール，構成管理ツール，デバッグ・テストツール）である」

これに対して，5.3 節の (2)，(3)，(7) の改善案が抽出されている．文献 10) では，プロジェクト管理ツール，構成管理ツール，デバッグ・テストツールを利用すると品質低下の防止に有効であると報告されている．我々が求めた改善案では，ドキュメント作成ツールを利用していない場合に文献 10) の分析結果の成り立つことが求まった．

このことは改善案が有効に機能する前提条件を明確にしたと解釈することもできる．企業で行われる様々な改善案が実際に想定したように機能しないことが起きる理由の 1 つには，こうした改善案の前提となる事項を見逃してしまうこともあると考える．そのような条件などを細かく抽出できる点も本手法の特徴といえる．

6. ま と め

本研究では，ソフトウェアの品質に関する指標「不具合工数密度」に関して，この指標を改善するための策を提案する手法を提案した．本研究は著者らが産学連携の共同研究として実施し，双方の知見を組み合わせることで実現できた．

具体的には，まず，相関ルールマイニングをプロジェクトデータに適用し，ソフトウェアの品質に関する相関ルール群を抽出した．次に，そのルール群について，類似のルールを抽出し，それらからプロジェクトの改善案を抽出することを試みた．企業横断的データを利用し，試作したシステムを用いて適用実験を実施した結果，多くの改善案をほぼ自動的に抽出することに成功し，改善案の検討が行いやすいことが確認できた．

なお，本手法は相関ルールの結論を変更することで，別の結論についても適用可能である．今後の課題として，コストや開発期間などプロジェクトの混乱状態に関係する別の結論についての実験なども行っていきたい．

参 考 文 献

1) Gupta, G., Strehl, A. and Ghosh, J.: Distance based clustering of association rules, *Intelligent Engineering Systems Through Artificial Neural Networks*, Vol.9, pp.759–764 (1999).
2) Michail, A.: Data Mining Library Reuse Patterns using Generalized Association Rules, *Proc. 22nd International Conference on Software Engineering*, pp.167–176 (2000).
3) Mizuno, O., Kikuno, T., Takagi, Y. and Sakamoto, K.: Characterization of risky projects based on project managers' evaluation, *Proc. 22nd International Conference on Software Engineering*, pp.387–395 (2000).
4) Morisaki, S., Monden, A., Tamada, H., Matsumura, T. and Matsumoto, K.: Min-

ing Quantitative Rules in a Software Project Data Set, 情報処理学会論文誌，Vol.48, No.8, pp.2725–2734 (2007).
5) Ohsugi, N., Monden, A., Kikuchi, N., Barker, M.D., Tsunoda, M., Kakimoto, T. and Matsumoto, K.: Is This Cost Estimate Reliable? – The Relationship between Homogeneity of Analogues and Estimation Reliability, *Proc. 1st International Symposium on Empirical Software Engineering and Measurement*, pp.384–392 (2007).
6) Takagi, Y., Mizuno, O. and Kikuno, T.: An Empirical Approach to Characterizing Risky Software Projects Based on Logistic Regression Analysis, *Empirical Software Engineering*, Vol.10, No.4, pp.495–515 (2005).
7) Yourdon, E.: *Death March: The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects*, Prentice-Hall Computer Books (1997).
8) Zimmermann, T., Weißgerber, P., Diehl, S. and Zeller, A.: Mining Version Histories to Guide Software Change, *IEEE Trans. Softw. Eng.*, Vol.31, No.6, pp.429–445 (2005).
9)（独）情報処理推進機構ソフトウェア・エンジニアリング・センター：ソフトウェア開発データ白書 2006，日経 BP 社 (2006).
10) 古山恒夫，菊地奈穂美，安田　守，鶴保政城：ソフトウェア開発プロジェクトの遂行に影響を与える要因の分析，情報処理学会論文誌，Vol.48, No.8, pp.2608–2619 (2007).
11) 水野　修，安部誠也，菊野　亨：プロジェクト混乱予測システムのベイズ識別器を利用した開発，*SEC journal*, Vol.1, No.4, pp.24–35 (2005).
12) 大杉直樹，角田雅照，門田暁人，松村知子，松本健一，菊地奈穂美：企業横断的収集データに基づくソフトウェア開発プロジェクトの工数見積り，*SEC journal*, Vol.2, No.1, pp.16–25 (2006).
13) 浜野康裕，天嵜聡介，水野　修，菊野　亨：相関ルールマイニングによるソフトウェア開発プロジェクト中のリスク要因の分析，コンピュータソフトウェア，Vol.24, No.2, pp.79–87 (2007).

(平成 19 年 11 月 11 日受付)
(平成 20 年 5 月 8 日採録)

出張　純也

平成 19 年大阪大学基礎工学部情報科学科卒業．現在，大阪大学大学院情報科学研究科博士前期課程 2 年．ソフトウェア開発プロジェクトのデータに対する定量的な分析に関する研究に従事．IEEE 学生会員．

水野　　修

平成 10 年大阪大学大学院博士前期課程修了．平成 11 年 3 月に同大学院博士後期課程中退．博士（工学）．同年 4 月より大阪大学大学院基礎工学研究科助手．平成 14 年 4 月改組により大阪大学大学院情報科学研究科助手．現在に至る．主にソフトウェア開発プロセスの改善支援，ソフトウェアのバグ検出手法に関する研究に従事．電子情報通信学会，IEEE 各会員．

菊野　　亨（フェロー）

昭和 50 年大阪大学大学院博士課程修了．工学博士．同年広島大学工学部講師．同大学助教授を経て，昭和 62 年大阪大学基礎工学部情報工学科助教授．平成 2 年同大学教授．現在，大阪大学大学院情報科学研究科教授．平成 20 年大阪大学留学生センター・センター長．主にフォールトトレラントシステム，ソフトウェア開発プロセスの定量的評価に関する研究に従事．電子情報通信学会，情報処理学会各フェロー．ACM，IEEE 各会員．日本信頼性学会理事．

菊地奈穂美（正会員）

昭和 60 年新潟大学理学部数学科卒業．平成 5 年 Stanford 大学コンピュータ・サイエンス修士（MSCS）．平成 18 年大阪大学大学院基礎工学部研究科博士後期課程修了．博士（工学）．沖電気工業（株）にて，ソフトウェアのプロセス評価・改善，メトリクス，設計・管理技術/ツール等の研究・技術移転と改善等に従事．平成 16 年 10 月より，独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター研究員を兼務．定量データ分析，見積手法および産学連携共同研究に従事．著作に『ソフトウェア開発データ白書』の監修等．PM 学会，IEEE 各会員．

平山　雅之

昭和 61 年早稲田大学大学院理工学研究科修了．昭和 61 年東芝入社．平成 15 年大阪大学大学院基礎工学研究科修了．博士（工学）．平成 17 年より IPA/SEC 組込みプロジェクト領域責任者兼任．平成 19 年東海大学専門職大学院客員教授．平成 20 年より情報処理学会監事．現在,（株）東芝ソフトウェア技術センター参事．組込みソフトウェアに関するエンジニアリング手法の研究と普及に従事．専門はソフトウェア品質・信頼性．日本品質管理学会各会員．