相関ルール分析を用いた
障害対応データの特徴分析

森崎修司，松村知子，玉田春昭，大杉直樹，門田暁人，松本健一
奈良先端科学技術大学院大学　情報科学研究科

概要

ソフトウェア開発プロジェクトにおいて，下流工程で発生する重大な障害は，開発コストの超過や納期の遅延を招く大きな原因となる．そのため，テスト工程での品質管理・進捗管理においては，問題の早期発見と対応が重要な課題となる．本稿では，重大な障害をプログラムの修正量・修正範囲などから判定しプロジェクトの遅延リスクを検出するために，プログラムの修正量・修正範囲の大小に影響を与える障害の特徴を抽出することを試みる．特徴の抽出には，EASE プロジェクトで開発した相関ルール分析ツール NEEDLE[2]を用いた．あるマルチベンダ情報システム開発プロジェクトにおける障害データに適用し，その結果を開発者に提示した結果，潜在的な修正量と障害の発見作業の関連が抽出できた．

1　はじめに

近年，障害管理情報は計算機上で管理され，未対応の不具合の列挙など自動化が進んでいる．本稿では，収集された障害管理情報から傾向や規則性を抽出し，類似や派生の開発に役立てることを目指す．具体的には，ソフトウェア開発の下流工程での障害内容と修正作業に関する情報から，修正に必要となった工数，ソースコードの修正量・修正範囲などの障害対応データの大小に影響を与える障害の特徴抽出を試行する．

特徴の抽出には，EASE(Empirical Approach to Software Engineering)プロジェクト[1]で開発した相関ルール分析ツール NEEDLE を用いた[2]．相関ルール分析は，「A(前提部)ならばB(結論部)である」というルールを抽出する分析方法である．NEEDLE では結論部の属性（修正工数，変更量）をユーザが指定し，対象データに含まれる規則性を前提部として列挙する．ルールの出現頻度や前提部(A) と結論部(B)の関連の強さを合わせて考慮することにより，回帰分析をはじめとする数式モデルにもとづく方法よりも，直接的なプロセス改善や計画立案支援が可能となる．

---

[1] http://www.empirical.jp/top.html

以降，2 章で提案する手法を述べ，3 章で適用事例を紹介する．4 章では，適用した結果得られた知見や問題点について説明し，5 章でまとめる．

2　相関ルール分析ツール NEEDLE

相関ルール分析は，「アソシエーションルール分析」とも呼ばれ，「ある事象が発生した時に，別な事象が発生する事後確率が高いような事象の組み合わせで，『属性 A を持つ対象は属性 B の生じる傾向がある』と一般的に記述される[3]．本分析手法の最も有名な例の１つとして，コンビニエンスストア等の小売店の購買履歴から相関ルールを抽出し，販売効率を大きくする商品の配置や料金の設定を行うものである．たとえば，コンビニエンスストアで同時に購入される商品の内容と曜日，時間帯を分析し，「休日に“レジャーシート”を買う顧客は“おにぎり”と“お茶”も同時に買っている」というルールを抽出できた場合，レジャーシートの配置をおにぎりかお茶に近づけ，発見率，併せ買い率を上げる，といった販売効率を大きくするような対応をとることができる．

EASE プロジェクトでは，この相関ルール分析をソフトウェアの開発プロジェクトのデータに適用する研究を行っている．ソフトウェア開発プロジェクトの特性をまとめたデータ（コスト、プロフィール、バグ）から規則性や傾向，例外を，アソシエーション分析手法を用いて抽出する機能を実現するツール NEEDLE を開発し[2]，様々なデータに適用し，成果を挙げている．例えば，プロジェクトのプロファイルデータからは「開発種別が拡張であり，かつ，アーキテクチャが 3 階層クライアントサーバアーキテクチャであれば，テスト工数比率が大きくなる」といったルールを抽出できた．このルールをベースに，3 階層アーキテクチャの機能拡張プロジェクトではテスト工数を大きめに見積る，といった対策を立てることが可能になる．

また，NEEDLE はソフトウェア開発データに多く含まれる数値データを処理するために量的ルール抽出機能を具備している．量的ルールは，結論部に量的変数を持つことができる．従来の相関ルール分析では，数値は大，中，小などの区間に離散化し，ルールを抽出するが，量的ルール抽出機能では結論部に数値をそのまま結論部として用い，結論部には平均値と標準偏差とする．たとえば，従来の相関ルールでは，障害の修正工数を区間であらわすことしかできなかったが，NEEDLE では，修正工数の平均値と標準偏差を得ることができ，区間よりもより多くの情報を得ることができる．量的ルール分析においては，ルール指標値として基準化平均（全体の平均に対して，ルールの前提部の条件に適合するケースの平均値の倍率），基準化標準偏差（全体の標準偏差に対して，ルールの前提部の条件に適合するケースの平均値の倍率）がある．基準化平均が非常に大きい（もしくは１未満で小さい），基準化標準偏差が小さいといった，従来

の相関ルール分析の区間からだけではわかりにくい情報を得ることができる.

3 分析事例

3.1 対象プロジェクト

本稿で分析対象としたプロジェクトは，今日の日本における情報システム開発の典型とも言える「マルチベンダ開発」である．具体的には，経済産業省の支援を受けて進められた中規模の情報システム開発プロジェクト「プローブ情報システム開発プロジェクト」を対象とした.

このプロジェクトでは，EASE プロジェクトとソフトウェアエンジニアリング技術研究組合(以降，COSE と呼ぶ)[2]，情報処理推進機構（IPA）下に設置されたソフトウェアエンジニアリングセンター（以降，SEC と呼ぶ）[3]の 3 者が連携して，データ収集体制の構築とデータ分析を行った．COSE はユーザ的立場の企業とプロジェクト管理担当企業，開発担当５社から成り，ユーザ的立場の企業が要求を出し，開発担当５社が分担してサブシステムの開発を行い，プロジェクト管理担当会社が５社を統括した全体のプロジェクト管理を行う.開発はウォーターフォールプロセスにより進められ，具体的には，要件定義を受けて基本設計を全社の協働で行い，以降各社で詳細設計，プログラム設計，製造，単体試験，社内結合試験が行われた．その後，各社のプログラムを集め，組合全体として社間結合試験，総合試験が行われた.

本プロジェクトは，2 年にわたって新規開発(一年目)，処理の共通化と機能追加（二年目）が行われたが，その 2 年目のデータを用いる.

3.2 収集データ

表 1　障害属性データ項目一覧

項目名	選択項目（カテゴリ）	項目説明
起票分類	- 仕様不具合 - コード不具合 - 仕様変更 - 不具合からの仕様変更	起票時の問題の種類
発見箇所	- 未入力 - 新規 - 改造(他システムからの流用) - 改造(システム内流用) - 再利用(未改造)	障害を発見した部分などの流用状況

[2] ソフトウェアエンジニアリング技術研究組合（COSE）：COnsortium for Software Engineering

[3] http://sec.ipa.go.jp/index.php

発見工程	- 未入力 - 要件定義 - 基本設計 - 詳細設計 - プログラム設計 - コーディング(製造) - 単体試験 - 結合試験１(コンポーネント内) - 結合試験２(コンポーネント間) - 総合試験 - 実証実験（運用）	障害を発見した工程
発見作業	- 未入力 - 分析・解析中 - システム動作中（テスト･運用） - レビュー	障害を発見した作業
試験項目種別	- 未入力 - 正常系 - 異常系 - 非機能 - その他	障害を発生した試験項目の分類
障害処理機能	- 未入力 - 入力データチェック機能 - 演算機能 - データ編集機能 - ファイル更新機能 - データ出力機能 - 連動（組み合わせ）処理 - 限界処理 - 外囲条件異常検知機能 - その他	障害を発見したシステムの機能
ソフトウェアエラー分類	- 未入力 - インタフェースエラー - 論理エラー - データ定義エラー - テーブル定義エラー - 形式エラー - その他	ソフトウェア上のエラー（誤り）の種別
重要度	- 未入力 - 重大 - 中度 - 軽微	障害の重要度
優先度	- 未入力 - 高 - 中 - 低	障害解決優先度
再現度	- 未入力 - 再現頻度大 - 再現度小 - 再現なし - 不明(未確認)	故障発生の再現頻度
障害原因	- 未入力 - 仕様書記述漏れ - 仕様書記述誤り - 仕様書記述不明確（曖昧）	故障発生原因の種類 仕様変更理由の種類

	- 仕様書記述標準違反 - 仕様書修正漏れ - 仕様書間不整合 - 仕様からの展開時の見落とし - 仕様の理解不足 - 仕様の確認不足 - 仕様書の検討粗漏 - コーディング時の言語用法の知識不足 - 再利用時のチェック漏れ - 修正時のチェック漏れ - 単なる凡ミス - 操作ミス - 周知連絡の不徹底 - 標準違反 - 指摘ミス(仕様どおり) - 原因不明 - その他	
エラーを作りこんだ工程	発見工程と同じ	障害発生の原因となった工程
障害を本来発見すべき工程	発見工程と同じ	障害を発見しなければならなかった設計もしくはテスト工程
障害を発見できなかった要因	- 未入力 - レビュー未実施(プロジェクト内レビュー/デザインレビュー/お客様レビュー) - レビュー指摘もれ - 再レビュー及び修正･確認もれ - 工程間引継ぎコミュニケーション不足 - 試験項目抽出もれ - テストそのもののもれ - 環境上の障害で後工程にもっていった - 結果確認ミス - その他	障害を発見しなければならなかった設計もしくはテスト工程で，摘出されなかった要因

本稿で分析するデータは，対象プロジェクトで用いられる障害管理ツール(GNATS[4])と構成管理ツール(CVS[5])から収集する．障害管理ツールと構成管理ツールから上記のデータを獲得するためのツールとして，著者が所属する EASE プロジェクトによって開発された EPM ツール[1]を用いる．このツールは，障害管理ツールや構成管理ツールからデータを抽出し，それらの加工データを図表に可視化したり CSV 形式で出力したりする機能を持つ．

本事例で用いたデータは，障害管理ツールや構成管理ツールから EPM が収集

[4] GNATS(GNU Bug Tracking System)：http://www.gnu.org/software/gnats/
[5] Concurrent Versions System　http://www.nongnu.org/cvs/

し xml 形式で出力する標準エンピリカルデータ[6]（以降，EPM データ）である．EPM データは，障害データに関しては障害管理ツールに入力されるすべての障害項目である（主な項目は，表 1 参照）が，構成管理ツールからは主に次のデータを出力する．

- CVS 上の操作日時
- 操作種別（ファイル追加，ファイル変更，ファイル削除，ファイル/ディレクトリの取り出し，など）
- 操作対象ファイル名
- 操作者
- (ファイル更新の場合)追加行数
- (ファイル更新の場合)削除行数
- 変更後ファイル行数
- 更新時コメント

データ収集にあたっては，各障害管理データと修正履歴データを関連付けて取り出せるように，開発開始前に障害管理ツールと構成管理ツールの運用ルールとして以下の依頼を開発担当者に行った．下記の運用ルールが守られたデータに関してのみ，障害管理データと障害対応データが 1 対 1 で関連付けることが可能になり，本研究の分析対象となる．

- 障害 1 件に対して，更新は 1 回（複数ファイルの場合も一括で更新）で行い，その際にコメントとして障害番号を入力する
- 複数の障害を一括で更新する場合は，対応した障害番号をコメントに列挙する
- 障害以外の更新に関しては，「未登録」というコメントを記入する

表 2 に各社からの収集データ数を示す．それぞれ，社内単体試験，社内結合試験での全障害数と障害管理データと障害対応データが 1 対 1 で関連付けられた修正量取得障害数を示す．特に，単体試験での軽微な障害やコードレビューによる発見欠陥は，複数の障害を一括で更新することが多く，そのため D 社や E 社は対応の取れない障害が多かった．

[6] http://empirical.jp/~epm/beta/help094/index-j.htm

表 2　社別障害数（社内単体～結合試験）

	A	B	C	D	E
GNATS 登録数（全障害数）	31	26	109	110	173
修正量取得障害数	23	14	101	65	143

3.3 データ加工

相関ルール分析ツール NEEDLE への入力データとして，前提部となる説明変数群と結論部となる障害対応データを抽出する．前提部（説明変数）として使用される障害属性項目は，表 1 のとおりである．障害管理データとしては，他に自由記述による問題詳細，修正方法，再現方法などがあるが，本研究では用いない．結論部となる障害対応データとしては，更新ファイル数・追加行数・削除行数を算出した．各データの定義は以下のとおりである．

- 更新ファイル数：１障害を修正するために更新したファイル数
- 追加行数：１障害を修正するために更新した各ファイルの追加行数の合計（ファイルを追加した場合は，追加したファイル行数が用いられる．）
- 削除行数：１障害を修正するために更新した各ファイルの削除行数の合計（ファイルを削除した場合は，削除したファイル行数が用いられる．）

3.4 分析方法

本研究では，以下に示す３つの分析パターン（Ａ１）～（Ａ３）を，NEEDLE を用いて行った．

（Ａ１）更新ファイル数についての相関ルール分析（質的ルール）

『更新ファイル数』の大小を結論とする質的ルールの抽出．これは，前提部・結論部の変数すべてが，質的変数として扱われる．更新ファイル数は，１以下を「普通」，２以上を「多い」の２つに分類する．

（Ａ２）追加行数・削除行数についての相関ルール分析（量的ルール）

『追加行数』『削除行数』に対する量的ルールの抽出．量的ルールは，以下の図１のような形式で結果出力される．

	A	B	C	D	E	F	G
1	ルール番号	ルール	支持度	平均	標準偏差	基準化平均	基準化標準偏差
2	1	(ADD_LINEの合計1＝mean61.538462;std84.541524)←(REMOVED-PHASE＝41.単体試験)＆(PRIORITY＝10.高)	0.2	61.53846	84.54152	2.001001	1.717708
3	2	(ADD_LINEの合計1＝mean61.538462;std84.541524)←(STATE＝90.完了)＆(REMOVED-PHASE＝41.単体試験)＆(PRIORITY＝10.高)	0.2	61.53846	84.54152	2.001001	1.717708
4	3	(ADD_LINEの合計1＝mean59.384615;std84.816998)←(UNDETECTED-CAUSE＝10.品質作り込み(品質項目反映)考慮不足)＆(PRIORITY＝10.高)	0.2	59.38462	84.817	1.930965	1.723305
5	4	(ADD_LINEの合計1＝mean59.384615;std84.816998)←(UNDETECTED-CAUSE＝10.品質作り込み(品質項目反映)考慮不足)＆(CLASSIFICATION＝20.コード不具合)＆(STATE＝90.完了)＆(PRIORITY＝10.高)	0.2	59.38462	84.817	1.930965	1.723305
6	5	(ADD_LINEの合計1＝mean59.384615;std84.816998)←(UNDETECTED-CAUSE＝10.品質作り込み(品質項目反映)考慮不足)＆(STATE＝90.完了)＆(PRIORITY＝10.高)	0.2	59.38462	84.817	1.930965	1.723305
7	6	(ADD_LINEの合計1＝mean59.384615;std84.816998)←(UNDETECTED-CAUSE＝10.品質作り込み(品質項目反映)考慮不足)＆(CLASSIFICATION＝20.コード不具合)＆(PRIORITY＝10.高)	0.2	59.38462	84.817	1.930965	1.723305
8	7	(ADD_LINEの合計1＝mean57.000000;std78.712134)←(REPEATABILITY＝10.再現頻度大)＆(CLASSIFICATION＝20.コード不具合)＆(STATE＝90.完了)＆(PRIORITY＝10.高)	0.246154	57	78.71213	1.853427	1.599267

図１　量的ルール抽出結果例

ルール番号１の例では，修正工程(REMOVED-PHASE)が単体試験で，優先度(PRIORITY)が高い障害の場合，追加行数の平均値が約 61 行で，標準偏差(ばらつき)は 84.54 となっている．基準化平均から，これは前提部が無い場合の平均値（つまり全データの平均値）の 2 倍程度であり，修正工程が単体試験で優先度が高い障害は，約 2 倍の行数が追加されることを示している．ただし，基準化標準偏差が約 1.7 ということで，ばらつきも大きくなっているため，必ずしもこのルールの適合性が良いとは言えない．(一部のデータの影響が大きいことが考えられる.)

（Ａ３）更新ファイル数についての例外的なルールの抽出

（Ａ１）で抽出した質的ルールに対して，結論部が逆転するルールを抽出する．例外的なルールは，以下，図２のように出力される．

1 行目の種類＝Basic が常識ルールである．それに対して，種類＝Exception が例外的なルールであることを示している．結論部が変化していることがわかる．種類＝Relation は，常識ルールに対して，例外的なルールに追加された条件のみを前提部としたルールであり，それぞれのルールの支持度・信頼度が示される．一般的に，例外的なルールは支持度や信頼度が高いわけではなく，意味のあるルールを抽出するには，Relation の条件の内容（重要な意味を持つ条件であるかどうか）や結論部の変化の度合い（小→中への変化より，小→重大）によって選択する．

本分析では,更新ファイル数が多い（2 以上）の常識ルールに対して，ある前

提条件の追加で更新ファイルが 1 つになる例外的なルール，または，更新ファイル数が普通（１つ）の常識ルールに対して，ある前提条件の追加で更新ファイルが 2 以上になる例外的なルールを抽出する

	A	B	C	D	E	F	G	H
1	一般ルール番号	例外ルール番号	種類	ルール	支持度	対一般ルール比	信頼度	リフト値
2	1		Basic	(PROJECT_IDのカウント＝1.[1.100000))←(INTRODUCED-PHASE＝40.コーディング(製造))＆(ERROR-CLASS＝00.未入力)＆(REPEATABILITY＝10.再現頻度大)＆(CLASSIFICATION＝20.コード不具合)	0.384615	1	0.932203	1.110876
3	1	1	Exception	(PROJECT_IDのカウント＝2.[1.100000,))←(INTRODUCED-PHASE＝40.コーディング(製造))＆(ERROR-CLASS＝00.未入力)＆(REPEATABILITY＝10.再現頻度大)＆(CLASSIFICATION＝20.コード不具合)＆(TEST-CLASS＝10.正常系)＆(UNDETECTED-CAUSE＝99.その他)＆(STATE＝90.完了)＆(REMOVED-PHASE＝40.コーディング(製造))＆(SEVERITY＝10.重大)＆(PREFERRED-PHASE＝40.コーディング(製造))	0.020979	0.054545	0.1875	1.165761
4	1	1	Relation	(PROJECT_IDのカウント＝2.[1.100000,))←(TEST-CLASS＝10.正常系)＆(UNDETECTED-CAUSE＝99.その他)＆(STATE＝90.完了)＆(REMOVED-PHASE＝40.コーディング(製造))＆(SEVERITY＝10.重大)＆(PREFERRED-PHASE＝40.コーディング(製造))	0.020979	0.054545	0.157895	0.981693
	1	2	Exception	(PROJECT_IDのカウント＝2.[1.100000,))←(INTRODUCED-PHASE＝40.コーディング(製造))＆(ERROR-CLASS＝00.未入力)＆(REPEATABILITY＝10.再現頻度大)＆(CLASSIFICATION＝20.コード不具合)＆(TEST-CLASS＝10.正常系)＆(UNDETECTED-	0.020979	0.054545	0.1875	1.165761

図 2　例外ルール抽出結果例

3.5 分析結果

ここでは，3.4 で示した３つの分析パターン（Ａ１）～（Ａ３）それぞれで抽出されたルール例を示す.

「（Ａ１）更新ファイル数についての相関ルール分析(質的ルール)」で抽出されたルール例

［A 社］

- 更新ファイル数が多い(2 以上)　→なし
- 更新ファイル数が普通(１)　　→
  発見できなかった要因＝レビュー指摘もれ　　＆
  エラーを作りこんだ工程＝コーディング（製造）　＆
  修正工程＝コーディング（製造）　　＆
  本来発見すべき工程＝コーディング（製造）

※更新ファイル数＝２（多い）は一件のみなので，注目すべきルールは出なかった

［B 社］

● 更新ファイル数が多い(2 以上) →
　　試験項目種別=異常系　　&
　　ソフトウェアエラー分類=インタフェースエラー　　&
　　発見工程=社内結合試験　　&
　　エラーを作りこんだ工程=プログラム設計　　【一部省略】

［C 社］
● 更新ファイル数が多い(2 以上) →
　　試験項目種別＝正常系　　&
　　再現度＝再現頻度大　　&
　　発見作業＝分析・解析中

［D 社］
● 更新ファイル数が多い(2 以上) →なし
● 更新ファイル数が普通(１) →
　　ソフトウェアエラー分類＝論理エラー　　&
　　問題処理機能＝データ編集機能　　&
　　起票分類＝コード不具合　　&
　　発見作業＝システム動作中（テスト・運用）

［E 社］
● 更新ファイル数が多い(2 以上) →なし
● 更新ファイル数が普通(１) →
　　エラーを作りこんだ工程＝コーディング（製造）　　&
　　問題原因＝単なる凡ミス

「（Ａ２）追加行数・削除行数についての相関ルール分析(量的ルール) 」で抽出されたルール例

［A 社］
● 追加行数の大きいルール→
　　エラーを作りこんだ工程＝コーディング（製造）　　&
　　ソフトウェアエラー分類＝インタフェースエラー　　&
　　発見工程＝単体試験　　&
　　重要度＝重大　　&
　　優先度＝高

● 削除行数の大きいルール→
　　ソフトウェアエラー分類＝インタフェースエラー　　&

問題処理機能＝データ出力機能　　&
重要度＝重大　　&
優先度＝高

※いずれのルールも基準化平均 2.7～4.1，つまり全体の平均値より， 2.7～4.1 倍の行数の追加／削除が行われている．

［B 社］

- 追加行数/削除行数の大きいルール→
  問題を発見できなかった要因=レビュー指摘漏れ　&
  優先度=高

※いずれのルールも基準化平均 2～2.5，つまり全体の平均値より 2～2.5 倍の行数の追加／削除が行われている．

［C 社］

- 追加行数の大きいルール→
  試験項目種別＝正常系　　&
  再現度＝再現頻度大　　&
  発見作業＝分析・解析中
- 削除行数の大きいルール→
  再現度＝再現頻度大　　&
  発見作業＝分析・解析中　　&
  発見箇所＝新規

※いずれのルールも基準化平均 2.5～3.5，つまり全体の平均値より 2.5～3.5 倍の行数の追加／削除が行われている．

［D 社］

- 追加行数の大きいルール→
  修正工程＝単体試験　　&
  優先度＝高
- 削除行数の大きいルール→
  本来発見すべき工程＝プログラム設計

※いずれのルールも基準化平均 2～3，つまり全体の平均値より 2～3 倍の行数の追加／削除が行われている．

［E 社］

- 追加行数の大きいルール→なし
- 削除行数の大きいルール→
  重要度＝重大　　&

本来発見すべき工程＝コーディング(製造)　&
優先度＝高　&
発見箇所＝新規

※基準化平均 2, つまり全体の平均値より 2 倍の行数の削除が行われている.

- 追加行数の小さいルール→
  エラーを作り込んだ工程＝コーディング(製造) &
  問題原因＝単なる凡ミス

- 削除行数の小さいルール→
  問題原因＝仕様確認不足 or 単なる凡ミス

※基準化平均 0.15～0.65, つまり全体の平均値より 0.15～0.65 倍の行数の追加/削除が行われている.

「（Ａ３）更新ファイル数についての例外的なルール抽出」で抽出されたルール例

［A 社］

- 更新ファイル数が普通（1）→多い（2 以上）
  *常識ルール**: 発見できなかった要因＝レビュー指摘もれ　&
  起票分類＝コード不具合
  例外的なルール: *常識ルール**　&
  再現度＝再現頻度大

※質的ルールと同じく，更新ファイル数＝２は一件のみなので，注目すべき例外的なルールは出なかった

［B 社］

- 更新ファイル数が普通（1）→多い（2 以上）
  *常識ルール**: エラーを作りこんだ工程＝コーディング（製造）　&
  重要度＝重大
  例外的なルール: *常識ルール**　&
  再現度＝再現頻度小　&
  問題処理機能＝連動（組合せ)処理　&
  本来発見すべき工程＝プログラム設計　【一部省略】

［C 社］

- 更新ファイル数が多い（2 以上）→普通（1）
  *常識ルール**: 試験項目種別＝正常系　&
  再現度＝再現頻度大　&

発見作業＝分析・解析中
例外的なルール：*常識ルール**　&
エラーを作りこんだ工程＝コーディング（製造）　&
ソフトウェアエラー分類＝論理エラー　&
重要度＝軽微【一部省略】

［D 社］

- 更新ファイル数が普通（1）→多い（2 以上）

*常識ルール**：ソフトウェアエラー分類＝論理エラー　&
再現度＝再現頻度大　&
重要度＝中度　&
発見作業＝システム動作中（テスト・運用）
【一部省略】
例外的なルール：*常識ルール**　&
試験項目種別＝正常系　&
問題処理機能＝データ出力機能　&
優先度＝中　&
発見箇所＝改造（システム内流用））

［E 社］

- 更新ファイル数が普通（1）→多い（2 以上）

*常識ルール**：エラーを作り込んだ工程＝コーディング(製造)　&
再現度＝再現頻度大　&
発見箇所＝新規
例外的なルール：*常識ルール**　&
発見工程＝コーディング(製造)　&
重要度＝重要　&
優先度＝高　【一部省略】

4　考察

（1）障害対応データに対する相関ルール分析の適用可能性

障害管理データ・障害対応データを用いて，質的ルール，量的ルール，および例外的なルールの抽出が可能であることは確認できた．データ数について社毎にかなりのばらつきがあったり，目的・説明変数によっては値に偏りがあったりしたが（すべての障害について修正量が入手できたわけではないため），それぞれのデータセットで何らかの分析結果を取得することができた．例外的なルールに関して，処理の完了に長時間を要する（ルール抽出に必要な計算量が莫大である）という状況はあったが，設定や投入する基本（常識）ルールのフィルタリングなどによって，

処理時間を短縮させることが可能である.

（2）分析手法および抽出されたルールに対する開発担当者の評価

分析手法（相関ルール分析および NEEDLE）については，短時間（10～15 分）で各社担当者にその基本的な概念や機能を理解してもらうことができ，有用である，という評価を得ている．うち一社においては，EASE が開催する NEEDLE 説明会への技術者の派遣，社内におけるツール適用範囲の拡大が検討されている．

ルールに対してはプロジェクト管理の観点から有益なものがあるとの指摘がある．例えば，C 社においては，「「発見作業＝分析・解析中」時に追加行数・削除行数が多くなる」というルールが発見されているが，そのような条件下で追加行数・削除行数が多くなる原因の一つとして，「障害の原因分析（デバッグ）中に「芋づる式」に同種のバグが見つかった.」ことのあることがインタビューで確かめられた．芋づる式にバグが見つかるという事態は，プロジェクト管理の観点からは見逃すことができないものであり，多発するようであれば，システム構造の見直しや再コードレビュー等の対策が必要である．今回発見されたルールに基づくプロセス監視技術の開発なども考えられる．

（3）社間共通ルールの抽出

3.5 で示したとおり，対象各社のデータからは，それぞれいくつかのルールを抽出することができた．但し，抽出されたルールの多くは，各社独自のものであり，全社に共通するルールは抽出されなかった．なお，全社のデータを対象に（全社のデータをひとまとめにして）ルール抽出を試みたが，3.4 で示した３つの分析パターン（Ａ１）～（Ａ３）では，有益なルールは得られなかった.

5　まとめ

本稿では，大きな障害をプログラムの修正量・修正範囲などから判定しプロジェクトの遅延リスクを検出することを目的として，プログラムの修正量・修正範囲の大小に影響を与える障害の特徴抽出を試みた結果について述べた．修正による更新ファイル数の特徴として，製造工程で発見/修正された障害は更新ファイル数が小さいという共通点が確認された．また，例外的なルールとして，プログラム設計工程（製造工程の 1 工程前）で発見することが本来できていたはずが実際には見落としていたものが，更新ファイル数が大きくなることを示すルールがみつかった．開発メンバへ抽出ルールを提示しながらインタビューしたところ相関ルールによる特徴抽出が今後のプロセス改善へとつながることを示す意見が得られた.

参考文献

[1] 大平 雅雄, 横森 励士, 阪井 誠, 岩村 聡, 小野 英治, 新海 平, 横川 智教: ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム, 電子情報通信学会論文誌 D-I, VolJ88-D-I, No.2, pp228-239, 2005.

[2] Shuji Morisaki, Akito Monden, Haruaki Tamada,Tomoko Matsumura, and Ken-ichi Matsumoto, “An extension of association rule mining for software engineering data repositories,” Nara Institute of Science and Technology Technical Report, TR2006008, (Nov. 2006)

[3] 山口和範，高橋淳一，竹内光悦著,「よくわかる多変量解析の基本と仕組み」，秀和システム，2004.